الفصل الأول والثاني

النصوص البرمجية من جهة المخدم

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

1. تعريف النصوص البرمجية من جهة المخدم
2. الاختلاف بين النصوص البرمجية من جهة المخدم والنصوص البرمجية من جهة الزبون
3. مفهوم مخدمات الوب ومخدمات الوب الأكثرانتشاراً
4. التقنيات و اللغات المستخدمة في النصوص البرمجية من جهة المخدم
5. كيفية إعداد مخدم IIS وإنشاء مجلدات افتراضية لاستضافة موقع

ما هو مخدم الوب؟

هو تطبيق مهمته استقبال طلبات مصدرها تطبيقات أخرى تدعى متصفحات الوب (أو زبون الوب)، وتقديم خدمة إرسال صفحات HTML مطلوبة من قبل هذه المتصفحات. يتم التواصل بين الزبون والمخدم اعتماداً على البروتوكول التطبيقي HTTP.

النصوص البرمجية من جهة المخدم

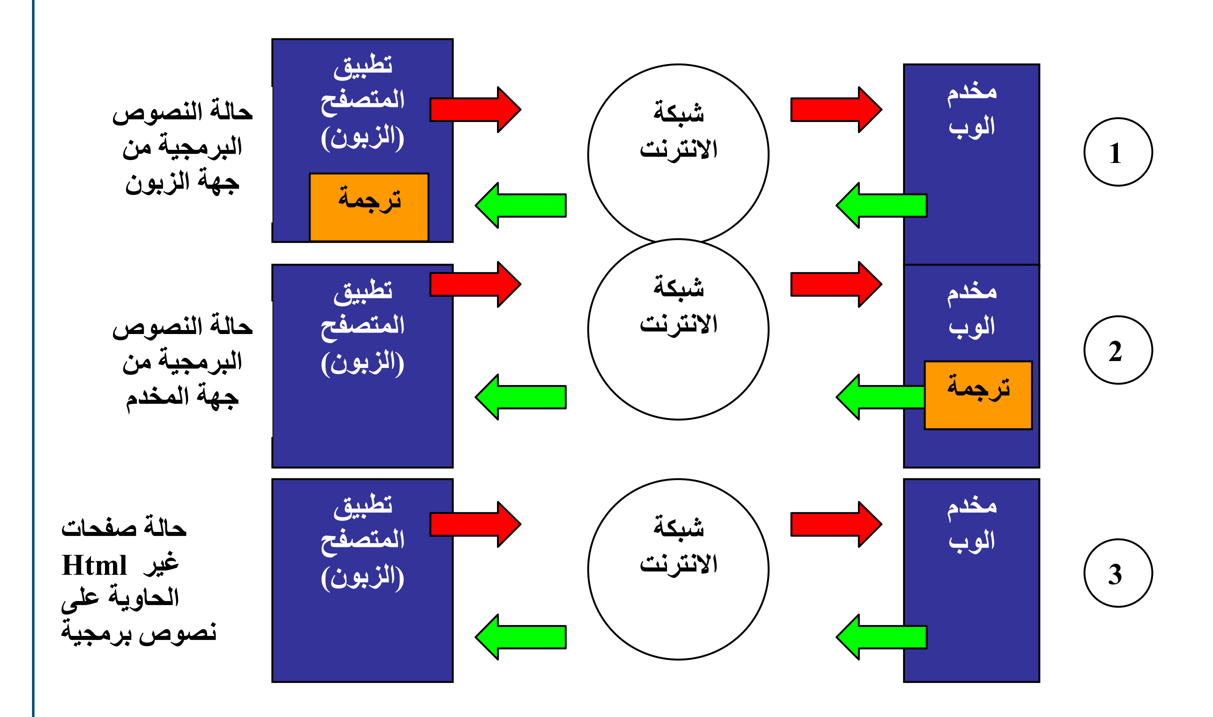
تعريف:

تُعتبر النصوص البرمجية من جهة المخدم إحدى تقنيات مخدمات صفحات الوب والتي يتم فيها الاستجابة لطلب المستخدم والتفاعل معه، عبر تشغيل نص برمجي على المخدم وتوليد صفحات HTML ديناميكية.

تُستخدم النصوص البرمجية من جهة المخدم عادةً، لتأمين تخديم مواقع الانترنت التفاعلية التي تشكل واجهة تعامل مع قواعد بيانات أو أي مصدر آخر للبيانات.

تم تمييز هذه التقنية بعبارة (من جهة المخدم) بسبب وجود تقنيات أخرى تكون فيها مسؤولية تنفيذ النصوص البرمجية على عاتق متصفح صفحات الوب (التطبيق الزبون)، وتسمى التقنية باسم النصوص البرمجية من جهة الزبون.

جهة المخدم أم جهة الزبون؟ (1)



لتوضيح فكرة النصوص البرمجية من جهة المخدم أو من جهة الزبون سنستعين بتمثيل بسيط لعلاقة مخدم الوب مع تطبيق زبون (المتصفح في حالتنا).

يمر التفاعل بين الزبون ومخدم الوب بالمراحل التالية:

1. يرسل التصفح (التطبيق الزبون) إلى المخدم طلب HTTP عبر الشبكة، بهدف استعراض صفحة معينة باستخدام محدد الموارد القياسي (URL).
2. يستلم المخدم الطلب ويبحث عن الملف المطلوب ليعيده عبر الشبكة إلى التطبيق الزبون.

تمثل الأسهم المتحركة باتجاه المخدم في الشكل الموضح ضمن الشريحة، طلب الزبون والذي يحدد فيه الصفحة التي يريد استقدامها من مخدم الوب، في حين تمثل الأسهم المتحركة باتجاه التطبيق الزبون استجابة مخدم الوب وإرسال المحتوى المطلوب إلى الزبون.

جهة المخدم أم جهة الزبون؟ (2)

نجد عند دراسة التفاعل بين مخدم الوب وزبون الوب، أننا أمام إحدى الحالات التالية:

الحالة 1 – تحتوي الصفحة على نصوص برمجية من جهة الزبون: تجري ترجمة وتنفيذ النصوص البرمجية المحتواة في الصفحة المطلوبة من قبل الزبون بعد استلامه للصفحة، نُطلق على هذه النصوص، النصوص البرمجية من جهة الزبون .

الحالة 2 – تحتوي الصفحة على نصوص برمجية من جهة المخدم: تجري ترجمة وتنفيذ النصوص البرمجية المحتواة في الصفحة التي أرسل الزبون في طلبها، من قبل المخدم قبل إرسال الصفحة، نُطلق على هذه النصوص، النصوص البرمجية من جهة المخدم .

الحالة 3– لا تحتوي الصفحة على أية نصوص برمجية (تحتوي على عبارات HTML فقط): يرسل المخدم الصفحة إلى الزبون الذي يستعرضها.

ملاحظة:
يمكن أن تحتوي الصفحة على نصوص برمجية من جهة المخدم، وعلى نصوص برمجية من جهة الزبون، يجري عندها التفاعل بين زبون ومخدم الوب وفق الحالتين الأولى والثانية معاً.

ما أهمية النصوص البرمجية من جهة المخدم؟ (1)

للإجابة على هذا السؤال ينبغي علينا الدخول في مقارنة بسيطة بين النصوص البرمجية من جهة المخدم والنصوص البرمجية من جهة الزبون من خلال سرد أهم النقاط الإيجابية والسلبية لكلٍ منهما.

النصوص البرمجية من جهة الزبون:

النقاط الإيجابية:
- تُساعد في دعم الحركة على الموقع باستخدام DHTML.
- تتحمل جزء من حِمل المعالجة على المخدم حيث تتم عملية ترجمة وتنفيذ النص البرمجي على الحاسب الزبون.
- تؤمن التفاعل مع المستخدم دون الحاجة إلى إعادة الاتصال مع المخدم (مثال عملية التأكد من بعض أخطاء البيانات المُدخَلة في نموذج معين).

النقاط السلبية:
- لا تملك القدرة على الوصول إلى أي مورد من موارد المخدم، ولا إلى أي تطبيق من التطبيقات المتصلة به مثل تطبيقات إدارة قواعد المعطيات. إذ يكون المصدر الوحيد للبيانات التي يمكن للنصوص البرمجية من جهة الزبون الوصول إليها، هي المعلومات المحتواة في الصفحة نفسها والتي تحوي، النص البرمجي، أو دخل المستخدم، أو معلومات من ملفات على جهاز الزبون (في حال تم منح الصلاحية للوصول إلى تلك المعلومات عن طريق المتصفح).
- توجد عدة لغات برمجة مُستَخدَمة في كتابتها، مما يعزز فرص عدم توافقيتها الكاملة مع كافة المتصفحات ويجعل بعض المتصفحات غير قادرة على تصفحها.
- لايتوفر الأمان الكافي عند استخدامها، إذ يمكن للزبون أن يستعرض بسهولة محتوى النصوص البرمجية من جهة

الزبون لأنها تكون جزءاً من النص المصدري (الذي يمكن استعراضه بالخيار view source من خيارات المتصفح) مما يجعل عملية استخدام أي نوع من التحقق أوكلمات السر من جهة الزبون غير آمن بالشكل الكافي.

ما أهمية النصوص البرمجية من جهة المخدم؟ (2)

النصوص البرمجية من جهة المخدم:

النقاط الإيجابية:

- تستطيع –كونها تعمل من جهة المخدم– الوصول إلى موارد المخدم والتطبيقات المرتبطة به مثل تطبيقات إدارة قواعد المعطيات.
- يولد تطبيقها صفحات HTML قياسية يستطيع أي متصفح تفسيرها واستعراضها أياً كانت لغة البرمجة المستخدمة في كتابة هذه النصوص البرمجية.
- لا يستطيع الزبون استعراضها لأن ما يصل للمستخدم هو نص HTML ناتج عن تفسير وتشغيل النصوص البرمجية، مما يجعل محتوى النصوص البرمجية من جهة المخدم أكثر أماناً.

النقاط السلبية:

- تُعتبر عملية التفاعل في النصوص البرمجية من جهة المخدم بطيئة لأنها تتطلب الاتصال بالمخدم عند كل تفاعل.
- يتحمل المخدم عبء عمليات ترجمة هذه النصوص البرمجية.

التقنيات المستخدمة في النصوص البرمجية من جهة المخدم

تاريخياً، جرى تطوير النصوص البرمجية من جهة المخدم، اعتماداً على لغات البرمجة التقليدية أو اللغات الخطاطية التقليدية مثل C و Perl و Shell script، بحيث كان التطوير يعتمد على إنشاء ما يسمى بـــ (CGI (Common gateway interface لتحصيل المعطيات التي يرسلها الزبون وللتعامل معها. وقد كان نظام التشغيل الذي يعمل عليه مخدم الوب، هو المسؤول عن عملية تنفيذ هذه البرامج بحيث كان يجري توجيه نتائج التنفيذ إلى تطبيق مخدم الوب.

جرى حديثاً تطوير تقنيات أخرى، كتلك التي تعتمد لغات مثل ASP و PHP، بحيث يجري تنفيذها مباشرةً من قبل مخدم الوب أو عن طريق وحدات برمجية إضافية وضمن فضاء العمل والعنونة المُخصصَين لمخدم الوب.

من أشهر التقنيات المُستَخدَمة في تطويرالنصوص البرمجية من جهة المخدم:

- PERL: لم يجر تصميم هذه التقنية لكتابة النصوص البرمجية الخاصة ببيئة الوب، ولكنها أثبتت فعالية عالية في هذا المجال، وهي ما تزال مستخدمة لكتابة برامج CGI والوحدات الخاصة بمخدم الوب APACHE نظراً لإمكانياتها الكبيرة في معالجة سلاسل المحارف، وهي العناصر التي يتم تبادلها بشكل أساسي في تطبيق وب. تأخذ

ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة PL.

- ASP: جرى تطوير هذه التقنية من قبل شركة مايكروسوفت، وهي تستخدم لغات مثل Java script و VB Script. تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة ASP.

- ASP.NET: جرى تطوير هذه التقنية أيضاً من قبل شركة مايكروسوفت، وركزت على اعتماد البرمجة المقادة بالأحداث، واعتماد إطار عمل "NET.". تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة ASPX.

- PHP: جرى تطويرها كتقنية من تقنيات المصادر المفتوحة (Open source)، وهي تكافئ تقنية ASP من حيث إمكانياتها، مع تمتعها بميزة الإنفتاح وبإمكانية التطوير والتحسين والتوسع من قبل العديد من الجهات نظراً لكونها مفتوحة المصدر. تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة PHP.

- ColdFusion: وهي نسخة تجارية من التقنية التي طورتها شركة Macromedia لدعم النصوص البرمجية من جهة المخدم. تتوفر هذه التقنية على أكثر من بيئة عمل و تدعم التعامل مع أكثر من نظام إدارة قواعد بيانات. تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة CF.

- JSP: وهي تقنية مبنية على لغة جافا لبناء نص برمجي من جهة المخدم. تتوفر هذه التقنية على أكثر من بيئة عمل(Windows/Unix/Linux) . تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة JSP.

مخدمات الوب الأكثر انتشاراً

1. مخدم Apache:



2. مخدم IIS(Internet Information Services):



3. مخدم Sun Java MicroSystem Web Server:



4. مخدم Zeus:



مخدم Apache:

يعتبر مخدم Apache، نظام مفتوح المصدر قدمته Apache software foundation ويتوفر مع رمازه مجاناً على منصات عمل Linux، Unix، Novell، Windows.

يتمتع هذا المخدم بالكثير من الخصائص المميزة نذكر منها: دعمه للعديد من لغات البرمجة مثل perl، php، ودعمه للبروتوكولات الآمنةSSL و TLS، وتوفيره لإمكانيات التحكم بشكل صفحات الخطأ، بالإضافة إلى توفر رمازه على نحو مفتوح مما يسمح بتطويره وتحسينه ومعالجة ثغراته بصورة أفضل وأسرع. يعد هذا المخدم من أكثر مخدمات الوب شعبيةً وانتشاراً على الإطلاق بحسب إحصاءات عام 2005.

مخدم IIS(Internet Information Services):

جرى تطوير هذا المخدم من قبل شركة Microsoft و هو عبارة عن مجموعة من الخدمات المخصصة لبيئة الوب والتي تعمل على نظام التشغيل Windows. يتم توزيع هذا التطبيق حالياً كخدمة من نظم التشغيل Windows 2000 و Windows 2003 server و Windows XP pro

تتضمن النسخة الحالية IIS 6.0 خدمات FTP, SMTP, NNTP, HTTP/HTTPS. ويُعتبر مخدم IIS المنافس الأقوى لمخدم Apache من حيث الشعبية و الانتشار، ولكنه، وحتى نسخته الحالية، يعاني من بعض نقاط الضعف وخصوصاً من النواحي الأمنية. وقد جرى تجاوز العديد من الثغرات في النسخة التجريبية IIS 7.0 وجرى تصميمها على شكل وحدات ومكونات منفصل مما يضيف مرونة أكبر في التعامل مع هذه المكونات.

مخدم Sun Java MicroSystem Web Server:

جرى تطوير هذا المخدم من قبل شركة Sun MicroSystem يُدعى حالياً SUN ONE. يتميز هذا المخدم بخصائص أمان عالية، وبسهولة استخدام مما يجعله مخصصاً لتطبيقات العمل المتوسطة و الكبيرة.

يتوفر المخدم على أغلب منصات العمل وهو يعطي العديد من الميزات للتطبيقات التي تستخدم تقنيات JAVA و JSP كما يدعم تقنيات ASP و PHP و تقنيات CGI.

مخدم Zeus:

تم تطوير هذا المخدم من قبل شركة Zeus technology، وهو يحتل المرتبة الأولى من حيث السرعة منذ عشر سنوات، ويعمل على

منصات عمل UNIX بمختلف أنواعها.

تثبيت مخدم IIS

الخطوات التي تسبق عملية التثبيت:

1. التأكد من تغطية عتاديات المخدم للحاجات الدنيا لنسخة IIS التي نثبتها. مثلاً، يُنصح باستخدام مخـــدم بـــذاكرة أوليـــة 512 ميغابايت ومعالج P4 كحد أدنى، عند استخدام النسخة IIS 6.0،.
2. التأكد من تثبيت عائلة البروتوكولات TCP/IP وذلك عن طريق إعدادات خصائص الاتصال الشبكي.
3. إعداد القرص الصلب الخاص بمخدم IIS وتهيئته باستخدام نظام الملفات NTFS.

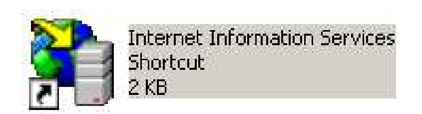
خطوات التثبيت:

تتضمن عدة إصدارات من نظام التشغيل Windows، المخدم IIS كإحدى خدماتها، لتثبيت هذا المخدم نتبع المراحل التالية:

1. نضغط زر Start ثم نختار خيار لوحة التحكم.
2. نختار خيار إضافة أو إزالة برامج ونحدد خيار إضافة أو إزالة مكونات Windows ونُفعّل خيار مخدم IIS.
3. يمكننا اختيار جزئيات التثبيت من خلال النقر على زر Details.

ملاحظة:

نلاحظ بالنقر على أيقونة Administrative tools من خيارات لوحة التحكم ظهور أيقونة خاصة للوصول إلى IIS.



كما نلاحظ أيضاً أنه يجري تلقائياً عملية إضافة مجلد باسم INETPUB ضمن القرص الصلب الذي يجري تثبيت IIS عليه.

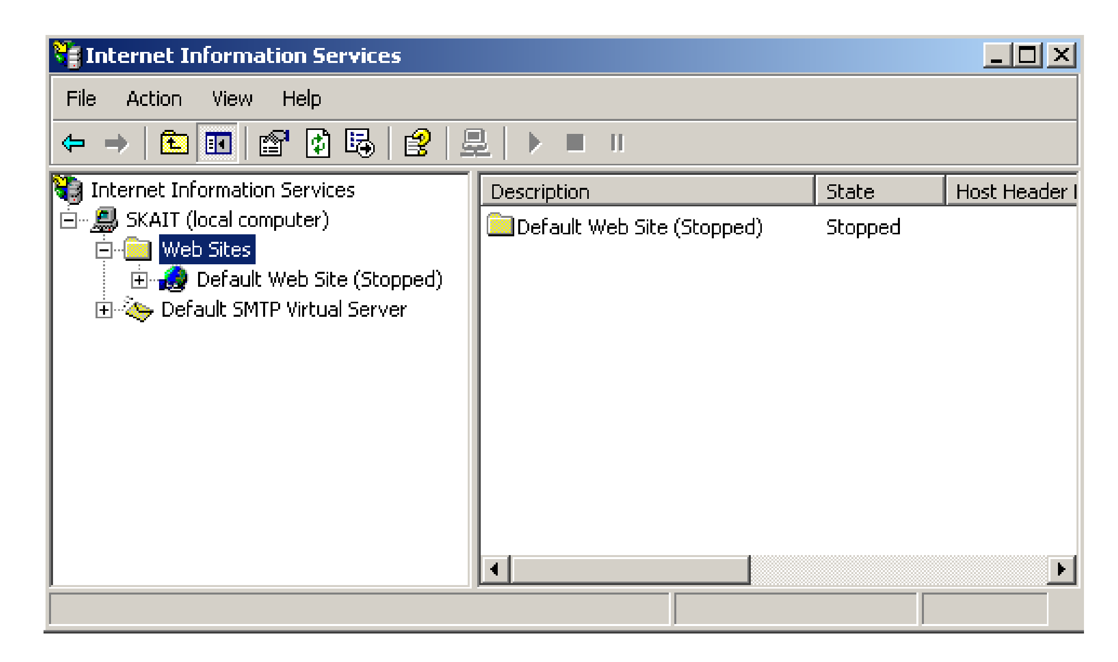
واجهة إدارة مجموعة مخدمات IIS

تتكون واجهة إدارة IIS، كما يظهر الشكل أدناه، من قسمين رئيسيين:

1. القسم الأيسر الخاص بإظهار مخدمات IIS العاملة على نفس الجهاز أو على عدة أجهزة ضمن الشبكة، بالإضافة إلى الخدمات

العاملة ضمن هذه المخدمات والمجلدات الرئيسية ضمنها.

2. القسم الأيمن الذي تظهر فيه الفهارس الفرعية والملفات المحتواة ضمن هذه المجلدات.



يقدم IIS خدمات متعددة، فهو يعمل كمخدم SMTP وكمخدم FTP إضافة إلى عمله كمخدم وب. كما يمكننا تفعيل وإيقاف تفعيل أي من الخدمات باستخدام خيارات Start – Stop – Pause عبر النقر بالزر الأيمن على أية خدمة من هذه الخدمات.

ملاحظة:

يمكننا ضبط الإعدادات التلقائية لمخدم وب IIS بالنقر على الخيار Property من قائمة شريط الأدوات.

استضافة مواقع الوب في IIS

تعتمد الفكرة الأساسية في استضافة المواقع على إنشاء مجلدات تحتوي وثائق وصفحات الـ HTML المراد نشرها، بالإضافة إلى النصوص البرمجية التي تعمل من جهة الزبون أو من جهة المخدم. يتولى مخدم IIS دور الوسيط الذي يستقبل طلبات متصفح الوب ويعيد المحتوى المناسب من المجلد المناسب.

يُنشئ IIS مجلداً تلقائياً باسم wwwroot يعتبر المجلد الرئيسي التلقائي. كما يقدم IIS آلية لإضافة مجلدات أخرى وإدارتها. حيث تشكل هذه المجلدات جذر شجرة المجلدات التي تحتوي وثائق وصفحات مواقع الوب.

يكفي لنشر صفحة وب تسميتها وفق إسم الصفحة التلقائية المُحدد ضمن إعدادات مخدم الوب، ونقلها إلى المجلد التلقائي المُحدد ضمن نفس إعدادات المخدم، بحيث يمكن لمتصفح الوب في الوصول إليها واستعراضها تلقائياً عد اتصاله بالمخدم.

ماهي المجلدات الافتراضية؟

لتمكين وصول مستخدمي المتصفح إلى صفحات موقع غير محتواة ضمن المجلد الرئيسي التلقائي يمكننا إنشاء ما يسمى بالمجلد الافتراضي.

يظهر المجلد للمتصفح وكأنه محتوى ضمن المجلد الرئيسي التلقائي بالرغم من كونه غير محتوى فيزيائياً ضمنه.

يمتلك كل مجلد افتراضي اسم بديل (أو اسم مسار)، وهو الاسم الذي يستخدمه المستعرض للوصول إلى هذا المجلد ويكون عادةً أقصر من المسار الفيزيائي الحقيقي لهذا المجلد.

تؤمن المجلدات الافتراضية عدة مزايا، نذكر منها:

1. درجة أمان أعلى كونها تحجب عن المتطفلين، المسار الفيزيائي الحقيقي للمجلد.
2. سهولة عملية إدارة المخدم، وجعل عملية نقل الملفات فيزيائياً –من مكان إلى مكان آخر على القرص– عملية سهلة لا تسبب أي تغيير للعناوين المستخدمة ضمن الصفحات.
3. سهولة استذكار المسار الافتراضي كونه أقصر من المسار الفيزيائي.

إضافة مجلد افتراضي لاستضافة موقع

لإضافة مجلد افتراضي بغرض استضافة مجموعة من صفحات الوب، ننقر بالزر الأيمن على خيار Default Website ثم نختار من القائمة New الخيار Virtual directory. تؤدي هذه العملية إلى تشغيل معالج خاص بإنشاء المجلدات الافتراضية.

1. يُطلب تحديد الاسم المُستَخدَم للوصول إلى هذا المجلد؛
2. يُطلب تحديد اسم الموقع الفيزيائي لهذا المجلد على القرص؛
3. يُطلب تحديد صلاحيات الوصول لمحتويات هذا المجلد.

بعد ضبط هذه الإعدادات يصبح بإمكاننا وضع ملف من نمط HTML ضمن المجلد الفيزيائي الحقيقي الذي يمثله المجلد الافتراضي ويصبح بإمكاننا استعراضه عن طريق متصفح الوب، حيث يمكننا الوصول إلى محتوى صفحة الوب مباشرة باستخدام محدد الموارد القياسي URL.

مثال:

يجري الوصول إلى صفحة test.html على مجلد افتراضي باسم myweb ضمن مخدم موصول على الشبكة ويحمل العنوان IP: 10.12.17.5، بكتابة محدد المورد القياسي:

http://10.12.17.5/myweb/test.html

فإذا كان لدينا مخدم DNS، نستطيع ربط http://10.12.17.5/myweb باسم نطاق معين فيصبح محدد الموراد القياسي من الشكل:

http://mydomain.com/test.html

النماذج في XHTML

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

1. لمحة عن XHTML
2. النماذج في XHTML
3. عناصر نماذج XHTML
   - الحقول النصية
   - حقول الاختيار
   - الأزرار
   - الحقول المخفية

ما هي XHTML؟

هو عبارة عن معيار متقدم عن المعيار HTML4.0 جرى وضعه لبناء نسخة HTML متوافقة تماماً مع الشروط الصارمة للمعيار XML من ناحية أسلوب كتابة التأشيرات، واحترام إغلاق كل تأشيرة مفتوحة، والتركيز على التداخل الصحيح للتأشيرات، وعدم التسامح في تجاوز بعض التأشيرات، واستخدام الفواصل: (" ") لتحديد قيم الواصفات مثل Attribute=”Value”، وغيرها.

النماذج في XHTML

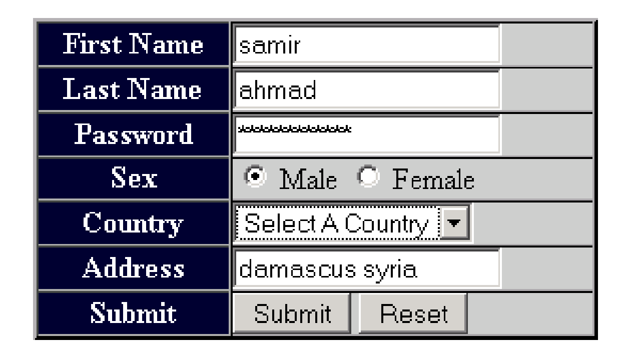
الغرض من النماذج:

1. تُعرَّف النماذج بأنها آلية تهدف إلى جعل صفحات HTML أكثر تفاعلية. وتسمح النماذج لمستخدمي صفحة الوب بتوزيع البيانات ضمن حقول محتواة في صفحة HTML وإرسال هذه البيانات إلى مخدم الوب حيث تجري معالجتها.
2. يتدرج تعقيد تفاعلية النماذج من حقل بحثٍ بسيط، إلى نموذج خاص بنظام بيع وشراء متكامل عبر الشبكة، أو إلى نظام استعلام إلكتروني، أو أية خدمة تمكِّن المستخدم من إدخال معلومات تمهيداً لمعالجتها.

مكونات النموذج:

1. يتكون النموذج من حقل إدخال أو أكثر. يمكن أن تكون هذه الحقول، حقول إدخال نصية، أو أزرار، أو مربعات اختيار ، أو قوائم، أو حتى خرائط صور.
2. تكون عناصر النموذج محصورةً بين التأشيرتين: <form> </form>.
3. يحتوي النموذج مكونات HTML أخرى، فهو لايقتصر بالضرورة على عناصر النموذج. فعلى سبيل المثال، يمكن للنموذج أن يحتوي نصوصاً و صوراً (تساعد في شرح كيفية ملء حقول النموذج). كما يمكن للنموذج أن يحتوي على نصوص برمجية من جهة الزبون، مكتوبة بلغة JavaScript أو غيرها، تساعد في عملية تقييم البيانات.

عند إرسال النموذج للمعالجة، يُرسل المتصفح بيانات الحقول إلى المخدم لمعالجتها أو إلى عنوان بريد إلكتروني أو إلى نص برمجي من جهة الزبون. يُعبر الشكل التالي عن نموذج تقليدي:



عناصر النموذج: الواصفة Method

تحدد هذه الواصفة أسلوب إرسال البيانات وطريقة إرسال طلب HTTP. تُميِّز هذه الواصفة حالتين:

1. حالة نموذج يستخدم طريقة POST: حيث يجري توجيه المتصفح لإرسال البيانات إلى مخدم وب أو إلى عنوان بريد إلكتروني، ضمن أغراض خاصة يتضمنها الطلب HTTP المُرسَل إلى المخدم.
2. حالة نموذج يستخدم الطريقة GET: حيث يجري توجيه المتصفح لإرسال البيانات إلى مخدم وب أو إلى عنوان بريد إلكتروني، على شكل سلسلة محارف تضاف إلى المُحدد URL ضمن طلب HTTP المُرسَل إلى المخدم.

بالإضافة إلى طريقتي GET و POST هناك العديد من الطرق الأخرى لإرسال طلب HTTP مثل الطرق PUT، Delete، Trace، Connect، لكننا سنتعامل حصرياً مع الطريقتين GET و POST، وسنستعرض لاحقاً الفروق الأساسية بين هاتين الطريقتين.

عناصر النموذج: الواصفة Action

1. تحدد الواصفة Action للمستعرض العنوان الذي يجب أن تصل إليه البيانات المُرسَلة، إذ يمكن أن تكون القيمة المُسنَدة للواصفة، عنوان بريد إلكتروني، أو محدد URL لصفحة تحتوي على نص برمجي من جهة المخدم.
2. تتم عملية الإرسال، والوصول إلى العنوان المحدد في الواصفة Action، عند نقر زر الإرسال Submit.

مثال 1:

```
<FORM METHOD="POST" ACTION="SK@scs-net.org">
… Content
</FORM>
```

يقوم هذا النموذج، عند ضغط زر Submit، بحزم البيانات و إرسالها كرسالة بريد إلى عنوان البريد المحدد في المثال SK@scs-net.org.

مثال 2:

```
<FORM METHOD="POST" ACTION="getInfo.aspx">
… Content
</FORM>
```

يقوم هذا النموذج، عند ضغط زر Submit، بحزم البيانات وإرسالها إلى الملف getInfo.aspx على المخدم والذي يحتوي النص البرمجي الخاص بمعالجة البيانات المُرسَلة.

مثال 3:

```
<FORM METHOD="GET" ACTION="Test.php">
… Content
</FORM>
```

يقوم هذا النموذج، عند ضغط زر Submit، بإرسال إلى ملف Test.php وذلك بإضافة البيانات إلى محدد URL الخاص بالصفحة، فإذا كانت إحدى الحقول المراد إدخالها هي name وكانت قيمة الحقل هي sami، يظهر محدد URL عند الإرسال، بالشكل التالي:

```
http://myDomain.com/Test.php? name=sami
```

عناصر النموذج: الواصفة EncType

1. تحدد الواصفة EncType نمط الترميز المُستَخدَم عند إرسال بيانات النموذج. تأخذ هذه الواصفة قيمها من أنماط المعيار MIME، وهو معيار يوصِّف أسلوب ترميز البيانات، جرى تطويره لترميز البيانات المختلفة ضمن شكل نصي وباستخدام حروف الأبجدية.

2. يجري تحديد نمط الترميز على النحو التالي: MIMEType/MIMESubType. فعلى سبيل المثال يظهر نمط الترميز التلقائي والذي يُعبِّر عن القيمة التلقائية للواصفة EncType كما يلي:

```
<Form ENCTYPE=application/x-www-form-urlencoded>
```

حيث يصلح الترميز السابق لجميع نماذج الوب عدا تلك التي تتطلب إرسال ملفات إلى مخدم الوب، والتي يجري فيها استخدام القيمة:

```
<Form ENCTYPE=multipart/form-data>
```

عناصر النموذج: الواصفة AcceptCharSet

1. تحدد الواصفة AcceptCharSet قائمة المحارف المتاح استخدامها في النموذج، وهي واصفة ضرورية خصوصاً في النماذج متعددة اللغات. تكون القيمة التلقائية لهذه الواصفة UNKNOWN، التي تشير إلى استخدام نفس قائمة المحارف المستخدمة في ترميز النموذج.

2. عند الحاجة لاستخدام أكثر من قائمة محارف، يمكن وضع قيم الواصفة السابقة على شكل قيم متتالية يجري فصلها بفاصلة، كما هو الحال في المثال التالي:

```
<Form AcceptCharset="windows-1256,iso-8859-1">
```

عناصر النموذج: الواصفة Target

تُستخدم الواصفة Target في بيئة الصفحات متعددة الأطر. تُعبر قيمة هذه الواصفة عن اسم الإطار الهدف الذي ستظهر فيه الإجابة بعد إرسال النموذج (في حال أعاد النص البرمجي من جهة المخدم أي خرج). فعلى سبيل المثال، تُستخدم هذه الواصفة عند الحاجة لإدخال بيانات خاصة بنموذج بحث متوضع ضمن إطار ما في صفحة HTML، وعند الحاجة لاسترجاع نتيجة البحث ضمن إطار آخر.

تأخذ الواصفة target القيم:

1. "_blank": تجري إعادة خرج النموذج في نافذة جديدة بدون اسم.
2. "_self": تجري إعادة خرج النموذج ضمن نفس الإطار الذي يحوي النموذج.
3. "_parent": تجري إعادة خرج النموذج ضمن الإطار الأب للإطار الذي يحوي النموذج.
4. "_top": تجري إعادة خرج النموذج ضمن الإطار الرئيسي مع إزالة كل الإطارات الأخرى.

مثال:

```
<form action="SomeThing.aspx" method="POST" Target="_blank" >
```

الأحداث ONSUBMIT و ONRESET:

1. يجري تنفيذ الحدث ONSUBMIT عند إرسال النموذج بضغط زر submit.
2. يجري تنفيذ الحدث ONRESET عند إلغاء معلومات النموذج بضغط reset .

مثال:

```
<form action="test.aspx"  method="POST" onsubmit= "window.alert('submitted successfully');">
```

تظهر، عند إرسال هذا النموذج، الرسالة "submitted successfully"

عناصر النموذج: واصفات أخرى

لا تقتصر واصفات وأحداث النماذج على تلك التي قمنا بشرحها في الشرائح السابقة، إذ توجد واصفات وأحداث أخرى سنذكر بعضها فيما يلي:

من هذه الواصفات:

1. ID: تُستَخدَم هذه الواصفة لتحديد اسم فريد للنموذج.
2. Class: تُستَخدَم هذه الواصفة في حال الرغبة باستعمال الأنماط الخاصة بملفات CSS المُعرَّفة ضمن التأشيرة Style من نفس الوثيقة.
3. Style: تُستَخدَم هذه التأشيرة لتعريف أنماط يجري استخدامها في تنسيق عناصر الوثيقة.
4. Title: تُستَخدَم لتحديد عنوان العنصر حيث يستفيد المتصفح من قيمة هذه الواصفة لإظهار مربع يحتوي هذه القيمة لدى مرور مؤشر الفأرة فوق العنصر.

ومن هذه الأحداث:

1. ONCLICK يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند النقر بالزر الأيسر فوق العنصر.
2. ONDBLCLICK يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند النقر المزدوج بالزر الأيسر فوق العنصر.
3. ONMOUSEDOWN يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند الضغط بالزر الأيسر فوق العنصر.
4. ONMOUSEUP يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند تحرير الضغط عن الزر الأيسر فوق العنصر.
5. ONMOUSEOVER يتم يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند وضع مؤشر الفأرة فوق العنصر.
6. ONMOUSEMOVE يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند تحريك مؤشر الفأرة فوق العنصر.
7. ONMOUSEOUT يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند خروج مؤشر الفأرة من المساحة التي يحددها العنصر.
8. ONKEYPRESS يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند ضغط وتحرير مفتاح فوق العنصر.
9. ONKEYDOWN يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند ضغط مفتاح فوق العنصر.
10. ONKEYUP يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند تحرير مفتاح فوق العنصر.

عناصر النموذج

مثال:

```
<form method=”POST” action=”Test.aspx”
target=”_self”  ID=”myForm”
class=”normalForm”  EncType=”multipart/form-data”  dir=”ltr”
onclick=”window.alert(‘you clicked the form’)”>
… Contents
</form>
```

1. يستعرض هذا المثال نص HTML يُعرِّف نموذج يعتمد على الطريقة POST في إرسال البيانات.
2. يحدد النموذج الملف test.aspx، كملف هدف يحتوي نصاً برمجياً من جهة المُخدم ويجري إرسال البيانات إليه للمعالجة.
3. تظهر الاستجابة على هذا النموذج (كنتيجة لتنفيذ الملف test.aspx) ضمن نفس الإطار الذي يظهر فيه النموذج، لأن النموذج استخدم القيمة ”_Self“ .
4. يمتلك النموذج السابق الإسم ”myForm“ ويجري تطبيق النمط ”normalForm“ على هذا النموذج (لا بد أن يكون هذا النمط معرف مسبقاً ضمن وثيقة CSS أو ضمن نفس الوثيقة باستخدام التأشيرة STYLE).
5. يُستخدَم في هذا النموذج الترميز ”multipart/form-data“ مما يعني أن النموذج يُرسل محتوى ملف إلى المخدم.
6. تظهر الرسالة ”you clicked the form“ عند النقر على النموذج.

حقول النماذج: الحقول النصية

يمكن استخدام عدة أنواع من الحقول النصية في النموذج نستعرضها فيما يلي:

حقل إدخال نص:

1. تُعتبر حقول إدخال النصوص من أكثر أنواع الحقول استخداماً في النماذج. يتألف حقل إدخال النص من حقل فارغ مخصص لإدخال سطر وحيد.
2. يجري تحديد حجم هذا الحقل باستخدام الواصفة Size، ويجري تحديد عدد المحارف المسموح إدخالها بالواصفة maxlength حيث تقاس Size و maxlength بعدد المحارف.
3. يمكن أن تكون الواصفة maxlength أكبر من الواصفة Size. في هذه الحالة يجري زلق محتويات الحقل النصي أثناء الكتابة.
4. يمكن استخدام الواصفة Value، لإعطاء قيمة تلقائية للحقل.

لا يوجد في HTML أية آلية مباشرة للتحكم بنوع المدخلات، لذا نلجأ إلى النصوص البرمجية من جهة الزبون، أو إلى النصوص البرمجية من جهة المخدم، لتقييم البيانات وإظهار رسائل الخطأ في حال عدم تطابقها مع نمط أوتنسيق البيانات المطلوبة.

مثال:

لإنشاء حقل نصي باسم mySample يحتوي على قيمة تلقائية هي test، ويكون بطول 10 محارف، وبعدد أقصى من المحارف يبلغ 40 محرفاً، نكتب النص التالي:

```
<input type="text" name="mySample" value="test" size="10" maxlength=40
/>
```

حقول النماذج: الحقول النصية

حقل إدخال كلمة السر:

1. يُستَخدَم هذا النمط من الحقول النصية كالحقول النصية العادية مع تطبيق قناع يخفي المحارف المُدخَلة. إذ لا يتمكن الناظر إلى الشاشة من رؤية المحارف، بل يرى عوضاً عنها محرف القناع الذي غالباً ما يكون أحد المحارف "O" أو "*"
2. يجري إظهار المحرف القناع ("O" أو "*") أثناء عملية الإدخال أمام الناظر إلى الشاشة، ولكن هذا لا يعني أن المحارف تكون مُقنعة بالنسبة للبرامج العاملة على الجهاز الذي تجري عليه عملية الإدخال أو أثناء عملية إرسال البيانات إلى المخدم.

مثال:

```
<input type="password" name="myPass" size="10" maxlength=25 />
```

حقول النماذج: الحقول النصية

حقل إدخال اسم ملف:

1. يسمح هذا النمط من حقول الإدخال النصية للمستخدم باختيار ملف حتى يجري إرساله إلى المخدم.
2. يكون اسم الملف محتوى في نص يعبر عن المسار الذي استخدمه المتصفح للوصول إلى الملف.
3. يقوم هذا النمط بإظهار الحقل النصي، إضافةً إلى زر يساعد في استعراض المجلدات بحثاً عن الملف المطلوب.
4. ليعمل هذا الحقل، لا بد من استخدام الطريقة POST في إرسال بيانات النموذج وعدم استخدام الطريقة GET.
5. يمكن استخدام الواصفة Accept مع هذا النمط للسماح بإرسال نمط معين من الملفات. تحدد قيم الواصفة Accept بنمط MIME للملفات المراد إرسالها.

مثال:

```
<input type="file" size=8 name="myFile" accept="text/*" />
```

في هذا المثال، يجري إنشاء حقل من نمط حقل ملف بإسم myFile، بحيث يكون عرض الحقل 8 محارف، وبحيث يقبل هذا الحقل الملفات من النمط Text بجميع أنماطه الفرعية مثل ر(text/html, text/css text/plain).

عناصر النماذج: الحقول النصية

حقل إدخال نص متعدد الأسطر:

يساعد هذا النمط من الحقول في إدخال نص متعدد الأسطر وذلك باستخدام التأشيرة <TEXTAREA>.

من أهم الواصفات الخاصة بهذه التأشيرة:

1. الواصفة Rows التي تحدد عدد الأسطر؛
2. الواصفة Cols التي تحدد عدد الأعمدة؛
3. الواصفة wrap التي تحدد قابلية النص للالتفاف عند تجاوز عرض مساحة النص. يمكن لقيمة الواصفة أن تأخذ إحدى القيم "off"، أو "virtual"، أو "Physical"، حيث تساعد القيمة "off" في تعطيل التفاف النص، بينما تساعد القيمة "virtual" في تثبيت التفاف النص ولكنها لا تقوم بإضافة أي فاصل إلى البيانات التي يجري إرسالها بعكس القيمة "Physical" التي تضيف فاصل سطر حقيقي عند كل التفاف للنص ويجري إرساله مع البيانات عند إرسال النموذج.

مثال:

```
<TEXTAREA rows="10" cols="6" name="myTextArea" wrap="off">
… This is a test
</TEXTAREA>
```

يُنشئ هذا المثال مساحة نصية بعرض 6 محارف وارتفاع 10 محارف باسم myTextArea ويجري في هذا النموذج تعطيل خاصة التفاف النص.

حقول النماذج: حقول الاختيار

يندرج تحت هذا التصنيف الحقول ذات الأنماط التالية:

مربعات التحقق:

1. يُنشئ هذا النمط من الحقول عنصر يمكن تفعيله أو إزالة تفعيله.
2. يكون الشكل التقليدي لهذا النمط عبارة عن مربع تظهر فيه إشارة X عند اختياره.
3. عند إرسال النموذج، يجري إرسال قيم العناصر المُختارة وهي قيم العناصر التي تظهر أمامها إشارة X. لذا يجب أخذ هذا الموضوع في الحسبان أثناء كتابة النص البرمجي الذي يعالج البيانات المُرسَلة.
4. تعتمد القيمة التي يجري إرسالها في هذا النمط من الحقول، على القيمة المُسندة للواصفة Value.
5. يمكن تفعيل حقل التحقق تلقائياً عند إظهار النموذج وذلك عبر إضافة الواصفة "Checked" إلى التأشيرة. لا تأخذ الواصفة "Checked" أية قيمة ولكن، للالتزام بقواعد كتابة الواصفات، نُسند لهذا النوع من الواصفات قيمة مساوية لإسم الواصفة ونكتبها: Checked="Checked"

مثال:

```
<input type="checkbox" value="test" name="myCheck"  Checked="Checked"
/>
```

حقول النماذج: حقول الاختيار

أزرار الراديو:

1. يشبه شكل هذا النمط من الحقول، شكل خيارات قنوات الراديو حيث يمكن اختيار خيار واحد فقط من الحقول التي تنتمي إلى مجموعة واحدة.
2. تعتمد القيمة التي يعيدها النموذج، كما هي الحال مع مربعات التحقق، على القيمة المُسندة للواصفة Value.
3. يمكن تفعيل حقل التحقق تلقائياً عند إظهار النموذج وذلك عبر إضافة الواصفة "Checked" وإعطائها القيمة "Checked" كما هو الحال في حالة مربعات التحقق.

مثال:

```
<input type="radio" name="myRadio" vaue="option1" checked="checked" />
first option<br>
<input type="radio" name="myRadio" vaue="option2" /> second option <br>
<input type="radio" name="myRadio" vaue="option3" /> third option
```

يظهر عند تنفيذ النص أعلاه النتيجة التالية، حيث يجري إرسال القيمة "first option"، وفقاً للاختيار المُبين في الشكل:

◉ first option
○ second option
○ third option

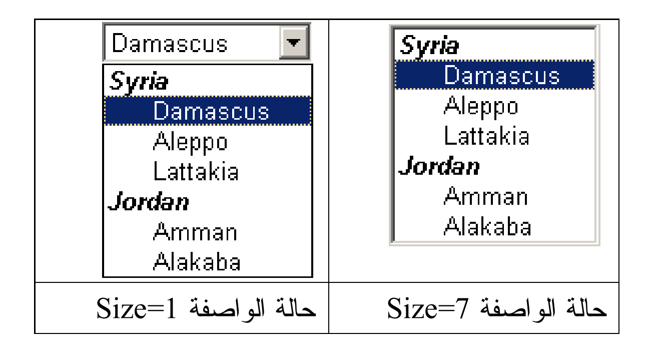
عناصر النماذج: حقول الاختيار

قوائم الاختيار:

1. تستخدم قوائم الاختيار التأشيرة <Select> لإنشاء نمطين أساسيين من القوائم:
   a. القوائم المنسدلة
   b. القوائم القابلة للزلق.
2. يساعد كلا النوعين في تنفيذ عملية اختيار خيار واحد أو عدة خيارات من قائمة خيارات.
3. يجري تحديد عناصر القائمة بالتأشيرة <option> حيث يجري تضمين كل خيار ضمن القائمة باستخدام التأشيرة <option></option>.
4. يجري إرسال المتحول المُحدد بالواصفة name، عند إرسال النموذج، محملاً بقيمة الخيار المحدد بـــ option في حال قام المستخدم باختيار وحيد. أما في حالة الاختيار المتعدد، فيجري تحميل متحول يمثل قائمة خيارات مفصولة عن بعضها البعض بفاصلة.
5. يكفي إضافة الواصفة multiple وإعطائها القيمة "multiple" لتحديد إمكانية استخدام عدة خيارات في قائمة الاختيار المعنية.
6. يستطيع المستخدم، عند تفعيل خاصة الاختيار المتعدد، اختيار أكثر من عنصر في القائمة بضغط زر CTRL والنقر على الخيارات المطلوبة. أما في حال عدم تفعيل هذه الخاصة، فإن قوائم الاختيار تعمل بنفس الآلية التي تعمل بها أزرار الراديو حيث تسمح باختيار خيار واحد فقط من مجموعة خيارات.
7. تتحكم قيمة الواصفة Size بارتفاع قائمة الاختيار القابلة للإنزلاق، وعندما تُسند إلى هذه الواصفة القيمة "1"، يجري إظهار سطر واحد من القائمة، وتتحول هذه القائمة إلى قائمة منسدلة.
8. لا توجد واصفة خاصة للتحكم بعرض قوائم الاختيار إنما تأخذ القائمة العرض اللازم لاحتواء الخيار صاحب العدد الأكبر من من المحارف.
9. يجري تحديد خيار تلقائي أومجموعة من الخيارات التلقائية باستخدام الواصفة Selected وبإعطاءها القيمة "Selected"، وذلك ضمن التأشيرة option لكل خيار من الخيارات المعنية.
10. يمكن إنشاء مجموعات فرعية من الخيارات ضمن قائمة الخيارات الواحدة وذلك باستخدام التأشيرة <Optgroup> وحصر جميع خيارت المجموعة الفرعية ضمن هذه التأشيرة.

مثال:

```
<select name="select">
<optgroup label="Syria">
  <option value="1">Damascus</option>
  <option value="2">Aleppo</option>
  <option value="3">Lattakia</option>
</optgroup>
<optgroup label="Jordan">
  <option value="4">Amman</option>
  <option value="5">Alakaba</option>
</optgroup>
</select>
```



حالة الواصفة Size=7	حالة الواصفة Size=1

عناصر النماذج: الأزرار

هناك ثلاثة أنواع أساسية للأزرار التي يمكن إضافتها إلى نموذج:

أزرار الإرسال و إعادة تأهيل النموذج:

1. يساعد زر الإرسال في إرسال بيانات النموذج إلى المخدم حتى تجري معالجتها عبر الملف المحدد في الواصفة Action الخاصة بالنموذج وذلك باستخدام الطريقة المحددة في الواصفة Method المرتبطة بالنموذج.
2. يستخدم هذا النمط الوسم input على أن يتم إسناد القيمة "Submit" إلى الواصفة type
3. يمكن استخدام التأشيرة button لتأدية نفس الوظيفة .

مثال:

```
<input type="submit" name="test" value="Send">
```

أو

```
<button type="submit" name="test" value="myValue">text on
button</button>
```

1. يؤدي زر إعادة تأهيل النموذج Reset إلى إفراغ الحقول من جميع القيم التي تم إدخالها من قبل المستخدم دون إرسال أية بيانات إلى المخدم.
2. يمكن استخدام التأشيرة input أو التأشيرة button لتحديد زر من هذا النمط بشرط إسناد قيمة "Reset" إلى الواصفة Type

مثال:

```
<input type="reset" value="reset">
```

أو

```
<button type="reset" >text on reset button </button>
```

عناصر النماذج: الأزرار

الأزرار من نمط صورة:

1. يُستخدم هذا النوع من الأزرار بهدف إضفاء جمالية على النموذج عبر تمثيل زر الإرسال بصورة .
2. يجري استعمال التأشيرة input مع إسناد القيمة "image" إلى الواصفة Type.
3. يجري تحديد ملف الصورة المُستخدَمة بإسناد المسار النسبي أوالمطلق لملف الصورة على المخدم إلى الواصفة Src.

مثال:

```
<input type="image" src="./images/myImage.gif">
```

عناصر النماذج: الأزرار

الأزرار متعددة الأغراض:

1. يساعد هذا النمط في إنشاء أزرار ذات أغراض متعددة مختلفة عما أوردناه حتى الآن.
2. يجري في هذا النمط من الأزرار استخدام الأحداث لتحديد الفعل ورد الفعل.
3. تستلزم هذه الأزرار استخدام نص برمجي من جهة الزبون.
4. تستخدم هذه الأزرار التأشيرة Input أو التأشيرة Button بعد إسناد القيمة Button إلى الواصفة type.

مثال:

```
<input type="button"
  value="Please Don't Press This Button"
  onclick="window.alert('any message.')" />
```

أو

```
<button type="button"
  value="anyValue"
  onclick="window.alert(any message.')"> 'Please do not press
 this button again.</button>
```

جرى في المثال السابق الإشارة إلى حدث الضغط على زر، بنص برمجي من جهة الزبون يقوم بإظهار رسالة. ويكمن الاختلاف الرئيسي بين Button و Input في أن التأشيرة Button تأخذ قيمة النص الموجود على الزر من النص المحصور بالتأشيرة، أما في حالة التأشيرة input فيتحدد النص الموجود على الزر بقيمة الواصفة value الخاصة بهذه التأشيرة.

عناصر النماذج: الحقول المخفية

الحقول المخفية:

1. تعد الحقول المخفية من العناصر الهامة في النماذج رغم كونها غير مرئية للمستخدم، حيث يهدف هذا النمط من الحقول إلى تمرير معلومات –عند إرسال النموذج– دون إظهارها في حقل واضح ضمن النموذج .

2. يُستخدم هذا النمط من الحقول عادة في حالتين:
   - لتمرير قيم إلى المخدم ليس لها أي دلالة للمستخدم، حيث يستفيد من هذه القيم النص البرمجي الذي يعالج المعطيات من جهة المخدم.
   - لتمرير معلومات يجب أن تبقى غير مرئية للمستخدم لأسباب تتعلق بسريتها. إذ يضمن إخفاء بعض المعلومات المُرسلة بهذه الطريقة حمايتها من المستخدم البسيط أو المبتدأ.

3. يستخدم هذا النمط من الحقول التأشيرة input بعد إسناد القيمة "Hidden" إلى الواصفة Type. أما القيمة التي يجري إرسالها عند إرسال النموذج، فهي القيمة المُسنَدة إلى الواصفة Value.

مثال:

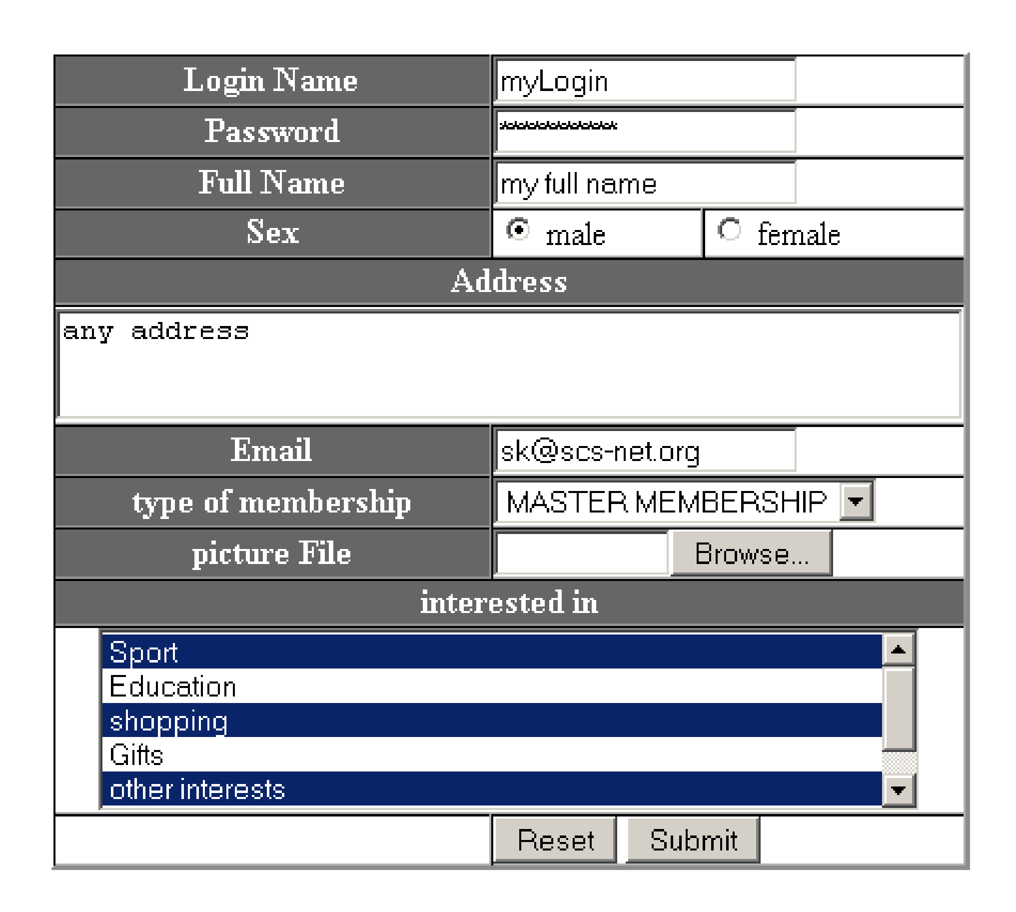
```
<input type="hidden" name="myHidden" value="value we want to hide" />
```

تكون أحد القيم التي تصل إلى المخدم، في حال جرى إرسال نموذج يحتوي النص أعلاه، هي القيمة:

myHidden=value we want to hide

تمرين عام

نريد إنشاء نموذج يعمل على جمع المحددة فيما يلي لإرسالها إلى ملف باسم register.aspx. نفترض أن النموذج يمتلك الشكل التالي:



الحل:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Test Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<style type="text/css">
<!--
.style1 {color: #FFFFFF}
-->
</style>
</head>

<body>
<form action="register.aspx" method="post" enctype="multipart/form-
data">
<table width="340" border="2" cellpadding="0" cellspacing="0"
```

```
bordercolor="#FFFFFF">
  <tr>
    <th width="185" bgcolor="#666666" scope="row"><span
class="style1">Login Name</span></th>
    <td colspan="2"><input name="login" type="text" id="login"></td>
  </tr>
  <tr>
    <th bgcolor="#666666" scope="row"><span
class="style1">Password</span></th>
    <td colspan="2"><input name="pass" type="password" id="pass"></td>
  </tr>
  <tr>
    <th bgcolor="#666666" scope="row"><span class="style1">Full Name
</span></th>
    <td colspan="2"><input name="fullName" type="text"
id="fullName"></td>
  </tr>
  <tr>
    <th bgcolor="#666666" scope="row"><span
class="style1">Sex</span></th>
    <td width="64"><input name="sex" type="radio" value="male">
      male</td>
    <td width="81"><input name="sex" type="radio" value="radiobutton">
      female</td>
  </tr>
  <tr>
    <th colspan="3" bgcolor="#666666" scope="row"><span
class="style1">Address</span></th>
    </tr>
  <tr>
    <th colspan="3" scope="row"><textarea name="address" cols="50"
id="textarea"></textarea></th>
    </tr>
  <tr>
    <th bgcolor="#666666" scope="row"><span
class="style1">Email</span></th>
    <td colspan="2"><input name="email" type="text" id="email"></td>
  </tr>
  <tr>
    <th bgcolor="#666666" scope="row"><span class="style1">type of
membership</span></th>
    <td colspan="2"><select name="select">
      <option>MASTER MEMBERSHIP</option>
      <option>NORMAL MEMBERSHIP</option>
      <option>GUEST MEMEBERSHIP</option>
    </select></td>
  </tr>
  <tr>
    <th bgcolor="#666666" scope="row"><span class="style1">picture
File</span></th>
    <td colspan="2"><input name="myPicture" type="file" id="myPicture"
size="10"></td>
```

```
  </tr>
  <tr>
    <th colspan="3" bgcolor="#666666" scope="row"><span
class="style1">interested in </span>            </th>
    </tr>
  <tr>
    <th colspan="3" scope="row"><select name="select2" size="5"
multiple>
      <option>Sport</option>
      <option>Education</option>
      <option>shopping</option>
      <option>Gifts</option>
      <option>other interests</option>

<option>___________________________________________________</option>
    </select></th>
    </tr>
  <tr>
    <th scope="row"> </th>
    <td colspan="2"><input type="reset" name="Submit2" value="Reset">
      <input type="submit" name="Submit" value="Submit"></td>
  </tr>
</table>
</form>
</body>
</html>
```

طلب HTTP باستخدام الطريقتين POST و GET

يتعرف الطالب في هذا الفصل على:

1. طلبات HTTP
2. طرق إرسال الطلبات
3. طريقتي GET و POST و الشروط المثلى لاستعمالهما
4. الطريقة HTTP GET والطريقة HTTP POST

ذكرنا في جلسة سابقة أثناء استعراضنا للتأشيرة "Form"، الواصفة "Method" و بيّنا إمكانية إسناد إحدى القيمتين "POST" أو "GET" لهذه الواصفة، وذكرنا أن هذه الواصفة تحدد كيف سيجري إرسال البيانات أو الطريقة التي ستُعتمد لإرسال طلب HTTP.

لفهم الفرق بين هذين الطريقتين لا بد لنا من شرح بسيط لأجزاء طلب بروتوكول HTTP.

البروتوكول HTTP:

يُعد البروتوكول HTTP البروتوكول التطبيقي الأساسي المُستخدم لتناقل البيانات على شبكة الأنترنت.

ويعتمد البروتوكول HTTP آلية (طلب/استجابة) بين التطبيق الزبون (المتصفح) والتطبيق المخدم (مخدم الوب)

طلب HTTP:

تبدأ مناقلة HTTP بإرسال طلب HTTP.

يتكون طلب HTTP من مجموعة من الأجزاء كما يظهر في الشكل التالي. سنستعرض فيما يلي هذه الأجزاء ونحدد لاحقاً نوع المعلومات التي يحملها كل جزء.

1	`GET / HTTP/1.1`
2	`Connection: Keep-Alive`
3	`Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*` `Accept-Language: en-us` `Accept-Encoding: gzip,`
4	`Content-type:application/x-www-form-urlencoded` `Content-length:23`
5	`Name=sami&type=3`

1	مُعرِّف (محدد) المصدر / الطريقة
2	الترويسة العامة
3	ترويسة الطلب
4	ترويسة الكيان
5	جسم الكيان

يتألف طلب HTTP مما يلي:

1. يتكون الجزء الأول من طلب http من العبارة (Method URI HTTP/version)، الذي يطلب الصفحة من المخدم عبر مُعرِّف المصدر (URI) باستخدام الطريقة Method. يُمثل القسم الباقي من هذا الجزء من الطلب (الجزء HTTP/1.1) نسخة البروتوكول المستخدمة.
2. يحتوي الجزء الثاني من الطلب الترويسة العامة ويحدد معلومات عامة كالتاريخ الحالي أو معلومات خاصة كالاتصال الحالي. يُعد هذا الجزء غير إجباري بحيث يمكن أن يتم إرسال طلب HTTP من دون هذا الجزء.
3. يحتوي الجزء الثالث الذي يدعى ترويسة الطلب على معلومات تتعلق بالزبون مُرسل الطلب و بنمط البيانات المُرسَلة و اسم المضيف. يُعد هذا الجزء غير إجباري بحيث يمكن أن يتم إرسال طلب HTTP من دون هذا الجزء.
4. يُستخدم الجزء الرابع الذي يدعى ترويسة الكيان، عند الشروع بإرساله وهو يحتوي معلومات تتعلق بنمط الكيان، وطوله، ومصدره و طريقة ترميزه. يكون هذا الجزء غير ضروري في حال عدم إرسال أي كيان.
5. يمثل الجزء الأخير جسم الكيان و يحتوي القيم التي يقوم الطلب بإرسالها مثل قيمة حقل ما.

طريقة GET أو POST

بالعودة إلى مخطط الأجزاء المكونة لطلب HTTP، نجد أنه يحتوي على الطريقة التي سيجري استخدامها في إرسال المعلومات. تأخذ هذه الطريقة إحدى القيمتين التاليتين:

GET: يجري طلب الوثيقة المحددة بالمحدد URL، ويجري إرسال بيانات المستخدم إلى العنوان المعين بالمحدد URL ضمن الترويسة نفسها.
POST: يجري طلب الوثيقة المحددة بالمحدد URL، ويجري إرسال بيانات المستخدم إلى العنوان المعين بالمحدد URL مع تضمين البيانات ضمن الجزء الخاص بجسم الكيان و ليس في أي من الترويسات.

تُستخدم هاتان الطريقتان أثناء العمل مع نماذج XHTML ، كما يدعم البروتوكول HTTP العديد من الطرق الأخرى نذكر منها:

HEAD: تتقدم بطلب مطابق للطلب الذي تتقدم به GET و لكن الجواب على هذا الطلب يتكون من ترويسة دون أي جسم. تفيد هذه الطريقة في الحصول على المعلومات الموجودة في ترويسة الجواب HTTP دون الحاجة إلى نقل كامل محتوى الجواب بما في ذلك نص الوثيقة التي يجري نقلها عادةً عند الإجابة على الطلب.
DELETE: تُستخدم لحذف الوثيقة المُشار إليها بالمحدد URL وهي نادرة الاستخدام.
TRACE: عند إرسال الطلب تجري إعادة نفس الطلب كاستجابة. بحيث تُمكن الزبون من معرفة المعلومات التي تضيفها المخدمات الوسيطة إلى الطلب أو معرفة أي تغيير يحصل على محتوى الطلب المرسل.
OPTIONS: تعيد قائمة بطرق HTTP التي يدعمها المخدم ويمكن استخدامها لاختبار عمل مخدم الوب.

GET أو POST

عند استخدام GET لإرسال بيانات النموذج من الضروري مراعاة مايلي:



1. يجب ألا يكون حجم هذه البيانات كبيراً؛
2. يجب ألا تحمل البيانات المُرسلة طابع السرّية.

تُستخدَم GET أيضاً بكثرة عندما تمرير قيمة معينة لملف عن طريق ارتباط تشعبي:

```
<a href=”test.aspx?x=10”>click here</a>
```

أما الطريقة POST فتستخدم لإرسال بيانات النموذج في حال:

1. أردنا إرسال بيانات بحجم كبير نسبياً؛
2. إذا كانت المعلومات التي يتعامل معها النموذج حساسة.

بعد أن تعرفنا على الفرق بين إرسال طلب HTTP باستخدام GET و إرسال طلب HTTP باستخدام POST، لا بد من توضيح سبب استخدام كلٍ من الطريقتين في نماذج XHTML:

تستخدم الطريقة GET لإرسال بيانات النموذج:

1. عندما لا يكون حجم هذه البيانات كبيراً نظراً لأنّ حجم البيانات الممكن إرسالها محدود بالحجم الأعظمي لمحدد المصدر، أي الـ URL
2. عندما لا تحمل البيانات المُرسلة طابع السرّية لأن هذه البيانات ستكون مكشوفة وستظهر ضمن حيز عنوان المستعرض
3. عند تمرير قيمة معينة لملف عن طريق ارتباط تشعبي

تُستخدَم GET بكثرة عند تمرير قيمة معينة لملف عن طريق ارتباط تشعبي. فعلى سبيل المثال، إذا أردنا تمرير القيمة X=10 إلى الملف test.aspx عند النقر على وصلة ما ولتكن الوصلة click here، تكون الصيغة على الشكل التالي:

```
<a href=”test.aspx?x=10”>click here</a>
```

أما الطريقة POST فتستخدم لإرسال بيانات النموذج في حال:

4. أردنا إرسال بيانات بحجم كبير نسبياً مثل ملف أو مربع نص يحتوي على حجم كبير من البيانات.
5. إذا كانت المعلومات التي يتعامل معها النموذج حساسة بحيث لايمكن إظهارها ضمن حيز العنوان في المستعرض.

الفصل الثالث والرابع

التعرف على البيئة

يتعرف الدارس في هذا الفصل على:

1. تقانات Microsoft المساعدة في تطوير مواقع وتطبيقات الو
2. بناء تطبيق وب باستخدام ASP.NET
3. طريقة التعامل مع معطيات التطبيق
4. التعامل مع مدخلات المستخدم ومع الأحداث

بيئة العمل Microsoft .NET

تمتلك بيئة العمل NET. بنية مكدس مؤلف من عدة طبقات من التقنيات التي تساعد في تطوير تطبيقات الوب أو أي نوع من التطبيقات الأخرى. كما تعتمد بيئة NET. البرمجية على العديد من المكتبات البرمجية (مثل System.Web، System.Data وغيرها) المكونة من آلاف الإجرائيات المكتوبة بلغات مثل ++C و #C و Visual Basic والمجهزة مسبقاً لتكون قابلة للاستخدام مباشرةً.

توفر بيئة العمل NET. بالإضافة للتقنيات التقليدية المستخدمة في بناء تطبيقات الوب (سواء كانت من جهة الزبون أو من جهة المخدم) تقنيات جديدة أو تقنيات محدثة عن تقنيات سابقة. نذكر بعضاً منها تعداداً على أن نعود إليها بالتفصيل:

1. ASP.NET 3.5
2. Web Services
3. JavaScript
4. ASP.NET AJAX
5. XML
6. Silverlight
7. Language Integrated Query (LINQ)
8. ADO.NET

بناء صفحة وب باستخدام ASP.NET

A. توليد موقع الوب:

1. تشغيل Visual Web Developer 2008 Express Edition.
2. توليد موقع الوب.
3. الحصول على البيئة البرمجية الازمة لبناء تطبيق الوب.

B. إضافة عناصر تحكم ASP.NET

نعرف عناصر التحكم بأنها أغراض تحتوي على رماز يمكن للمتصفح أن يفهمه ويقبله ويفسره عند فتحه للصفحة التي تحتوي هذه الأغراض.

1. لنذهب إلى زر Design لعرض صفحة التصميم والتي سيظهر فيها صندوق مظلل بالرمادي ومحاط بخط منقط. يشير هذا الخط إلى مقطع مكافئ لتأشيرة <div> في HTML.

2. اعتباراً من واجهة View→Toolbox اسحب أحد عناصر التحكم إلى داخل المستطيل الذي ذكرناه في الفقرة السابقة

C. مشاهدة صفحة ضمن المتصفح:

يمكننا مشاهدة صفحة ضمن المتصفح وهي في حالة البناء. إذ تقدم بيئة Net. مخدم وب يساعد في معالجة الرماز المصدري المكتوب بلغة ASP.Net وتوليد صفحات HTML. لتنفيذ ذلك يمكن للمطور:

1. وضع الفأرة ضمن صفحة ASP.Net؛

الضغط بالزر اليميني واختيار View in Browser.

إعداد البيئة

بعد إقلاع البيئة، يحصل المطور على الإعدادت التلقائية التي تقدمها البيئة، ولكن بإمكانه تغيير هذه الإعدادات بسهولة.

1. عرض الإعدادات:

للتأكد من أن الإمكانيات جميعها متاحة:

- اختر Tools→Options.
- في نافذة Options اختر Show All Settings.

2. لإلغاء بعض الإعدادات الخاصة بالمبتدئين والتي تخفي إمكانات متقدمة خاصة بلغات البرمجة:

- اختر Tools→Options.
- في نافذة Options افتح الخيار Text Editor واختر All Languages.
- في المنطقة الخاصة بـ Statement Area ألغِ الخيار Hide Advanced Member Box.
- ألغِ الخيار Single-click URL Navigation في أسفل صندوق الخيارات نفسه.

3. لجعل الصفحات تقلع اعتباراً من Design View:

- اختر Tools→Options.
- في نافذة Options اختر Show All Settings.
- اختر HTML Designer → General.
- اختر Design View.

إضافة عناصر تحكم باستخدام Toolbox

سنختار مجموعة من عناصر التحكم الخاصة بتقنيات AJAX لإضافتها كوننا سنستخدمها لاحقاً:

1. إذهب إلى الموقع www.codeplex.com/AtlasControlToolkit؛
2. اختر Downloads؛
3. قم بتنزيل الملف AJAXControlToolkitSource.zip (يمكن تنزيل الملف نفسه من المحتوى مباشرةً ودون الحاجة للذهاب إلى موقع الوب)؛
4. بعد تنزيل الملف قم بفك ضغطه ضمن مجلد C:\ACT؛
5. إذهب إلى ToolBox واضغط بالزر اليميني للفأرة على المنطقة الفارغة التي تلي General group؛
6. اختر Add Tab وأدخل الإسم AJAXControlToolkit؛

7. اضغط على AJAXControlToolkit ومن ثم بالزر اليميني على الفراغ الذي يليه واختر Add Items؛
8. اختر Browse من النافذة التي تظهر واذهب إلى مجلد: C:\ACT\SampleWebSite\Bin؛
9. اختر AjaxControlToolkit.dll واضغط OK.

ستظهر عناصر التحكم المطلوبة بشكل مختار يكفي الضغط على OK لتظهر تحت AJAXControlToolkit.

التعامل مع واجهة التطوير الأساسية

فيما يلي بعض أساليب التعامل مع واجهة التطوير التي تدعى Solution Explorer.

1. تنظيم الملفات في Solution Explorer

يعتبر Solution Explorer المكان الذي يجري فيه إضافة وتجميع الملفات الخاصة بتطبيق الوب. يمكن استخدام خاصة Drag&Drop بين Solution Explorer وبين نظام ملفات Windows لنقل ملفات من هذا الأخير باتجاه Solution Explorer.

فعلى سبيل المثال، يمكن بالنقر بالزر اليميني على Project Name (يدعى تلقائياً WebSite1) واختيار Add New Item أن تظهر لنا مجموعة الملفات التي يمكن اضافتها. ويمكن عندها اختيار XML File بحيث تظهر ضمن Solution Explorer حيث يمكن استخدامها في عملية التطوير البرمجي في حال كانت هناك حاجة لوجودها.

2. استخدام نوافذ وواجهات "الخصائص":

يتعامل المطور الذي يستخدم البيئة مع نافذة معممة على مختلف العناصر والتي تدعى نافذة الخصائص (Properties). تكون هذه النافذة عادةً مهمة جداً لإعطاء عناصر التحكم خصائص مختلفة تتناسب ونمط التطبيق المطور. فعلى سبيل المثال:

1. إذهب إلى صفحة Design View من الصفحة التي قمت بتوليدها.
2. اختر من Toolbox عنصر التحكم Button من فئة Standard وأنقله إلى الصفحة.
3. اضغط بالزر اليميني للفأرة على العنصر Button واختر Properties. ستظهر في أسفل النافذة من ناحية اليمين خصائص هذا العنصر.
4. يمكن بالضغط على أيقونة الترتيب الأبجدي في نفس مقطع Properties أن نرتب الخصائص ترتيباً أبجدياً.
5. بالذهاب إلى الخاصة ID يمكن تغيير قيمتها إلى btnAccept مثلاً واختيار لون بالنسبة للزر والخط وغيره.

بالذهاب إلى زر Source المجاور لزر Design View سيظهر الرماز الذي تم توليده بعد تغيير الخصائص

بناء موقع ASP.NET

سنقوم فيما يلي بتوليد موقع بسيط يتعامل مع جدول ضمن قاعدة معطيات SQL Server:

1. توليد الموقع:
   1. اختر File→New Web Site؛
   2. عند ظهور مربع الاختيار اختر Empty Web Site؛
   3. حدد مكان لتخزينه (مثل D:\MyWebSite)؛

4. يمكن اختيار لغة التطوير #C أو Visual Basic.
5. بهذا يكون لدينا موقع فارغ تماماً ويمكننا البناء اعتباراً منه.

6. استخدام SQL Server Express Database:
1. اضغط بالزر اليميني على الموقع واختر Add New Item؛
2. اختر SQL Database؛
3. ضع الإسم الذي تختاره لقاعدة المعطيات (MyWebSite مثلاً)؛
4. سيقوم بتوليد المجلد App_Data ووضع القاعدة ضمنه.
5. بالضغط ضغطتين على MyWebSite.mdf سيظهر Database Explorer.
6. بالضغط بالزر اليميني على Tables واختيار Add Table ستظهر واجهة تسمح بإضافة حقول للجدول وبإضافة الجدول ومفاتيحه وفقاً لخيارات المطور.
7. بالذهاب إلى File→Save All سيطلب منك النظام تسمية الجدول لحفظه.
8. إضغط بالزر اليميني على الجدول واختر Show data.
9. بالذهاب إلى النافذة اليسارية يمكنك ملئ الجدول.
10. اضغط بالزر اليميني على الموقع واختر Add New Item؛
11. اختر Web Form ستظهر Default.aspx؛
12. بعملية Drag & Drop خذ الجدول الذي قمت ببناؤه وأنقله إلى مقطع div الموجود في الصفحة Default.aspx. سيتم عرضه ضمنها.
13. يمكنك الآن عرضه عبر الضغط بالزر اليميني على الصفحة واختيار Show in Browser.

التعامل مع المعطيات وإدارتها ومعالجتها

سنستعرض في هذه الفقرة طريقة توليد وبناء قاعدة معطيات مرتبطة بتطبيق وب وطريقة التعامل مع مكونات هذه القاعدة. عموماً نحتاج لتأمين العمليات الأساسية التالية عند التعامل مع قاعدة معطيات:

1. إضافة معطيات إلى القاعدة؛
2. البحث عن معطيات موجودة ضمن القاعدة؛
3. تحديث معطيات ضمن القاعدة؛
4. حذف معطيات من القاعدة.

كما يمكن إضافة عمليات أخرى خاصة بالتطبيق الذي يتعامل مع قاعدة المعطيات وبحيث نجعل هذا التطبيق:

5. يقدم المعطيات بشكل جذاب وأنيق؛
6. يحسن من زمن جلب المعلومات وعرضها على الشاشة.

التعرف على Smart Tags

تمتلك عناصر تحكم ASP.NET المتقدمة مثل العنصر GridView أدوات مساعدة على تصميمها ندعوها Designers. تتمثل هذه الأدوات بمجموعة من الإجرائيات التي تساعد على إعداد هذه العناصر وفق خيارات متعددة. تظهر معظم أدوات التصميم عند استدعاء الواجهة Smart Tag وهي لوحة من الإجرائيات المساعدة على تصميم عنصر التحكم.

للوصول إلى هذه الواجهة يكفي إزاحة عنصر تحكم (زر من الأزرار) اعتباراً من Toolbox ومن ثم الضغط بالزر اليميني عليه وتأشير Show Smart Tag.

تحسين عنصر التحكم GridView

يمثل عنصر التحكم GridView واجهة للتعامل مع المعطيات الموجودة في القاعدة (الترتيب، الاختيار، العرض) ويعتبر من عناصر التحكم المهمة والفاعلة التي سنستعرضها بالتفصيل لاحقاً. في هذه المرحلة سنضع العنصر GridView على صفحة الوب ونقوم بتنفيذ بعض الإعدادات عليه:

1. اختر عنصر التحكم GridView من مجموعة الأدوات Data واضغط بالزر اليميني على زر Smart Tag.
2. اختر Auto Format. تعرض هذه النافذة الـ Schemas المتوفرة. لنختر (Professional) مثلاً.
3. اختر مصدر المعطيات (Data Source) وليكن متمثلاً بقاعدة المعطيات التي سبق وولدناها.
4. يمكنك الآن اختيار ... Sorting, Paging, والتي ستسمح بترتيب الأسطر ضمن GridView وتوزيع المعطيات على صفحات، والقيام بعملية اختيار لسطر من أسطر الــ Grid.
5. يمكن الآن وبعد العرض بالمتصفح أن نختار سطراً بالضغط على Select.
6. يمكن أيضاً أن نرتب وفق الترتيب الأبجدي في العمود First Name بالضغط عليه.
7. إذهب إلى Smart Tag وأضغط على Edit Columns ومن ثم اختر Selected Fields.
8. يمكنك الآن تغيير إعدادت الإظهار مثلاً الخاصة بحقل من حقول الجدول (لون الخلفية وغيره).

عنصر التحكم FormView

يساعد عنصر التحكم FormView في توليد المعطيات. لتنفيذ ذلك.

1. اختر عنصر التحكم Form View من مجموعة الأدوات Data واسحبه إلى أسفل GridView.
2. اختر مصدر المعطيات (Data Source) وليكن متمثلاً بقاعدة المعطيات التي اخترنا التعامل معها في GridView.
3. تساعد FormView على إضافة عمليات Insert, Update ... وغيرها اعتباراً من الواجهات Template.

تثبيت عنوان الصفحة

4. بالذهاب إلى Design View والضغط بالزر اليميني للفأرة على المساحة البيضاء واختيار Properties يمكن الذهاب إلى المربع اليميني الذي يحتوي على الخصائص عنوان الصفحة إلى Persons مثلاً. سيظهر عنوان الصفحة في أعلى المتصفح عند استعراضها باستخدام View in Browser.

التعامل مع الأحداث ومع إدخلات المستثمر

يمكن استخدام عنصر التحكم TextBox بأنواعه لإدخال المعلومات. كما يمكن استخدام عنصر التحكم RadioButton لخيار وحيد. واستخدام عنصر التحكم CheckBox لخيارات متعددة. كمثال سنقوم بإعداد تهيئة لا ئحة خيارات:

- اسحب عنصر التحكم Label إلى الصفحة وأسند القيمة lblPrompt للخاصة ID وأسند القيمة (Rate Your Fear of the Borg) للخاصة Text.
- اسحب عنصر التحكم RadioButtonList إلى الصفحة وأسند القيمة rblBorg للخاصة ID.
- أضف عنصر Label آخر مع إسناد القيمة lblResponse للخاصة ID.

- أضف عنصر Button.
- اضغط على Smart Tag الخاصة بالعنصر RadioButtonList واختر EditItems لإضافة لائحة من الخيارات.
- للحصول على نتيجة الخيارات التي سيستخدمها المستخدم، يمكن بالضغط المزدوج على الصفحة الفارغة فتح الصفحة الخاصة بالإجرائيات المرتبطة باللائحة. أضف ضمن الإجرائية Page_Load التعليمة: lblResponse.Text=rblBorg.SelectedValue.

الآن يمكن اعتباراً من مجموعة المعطيات ولائحة الخيارات التي أضفناها أن نحصل صفحة بالمتصفح حسب ما يظهر في الشكل.

الفصل الخامس والسادس

أساسيات نماذج الوب

يتعرف الدارس في هذا الفصل على:

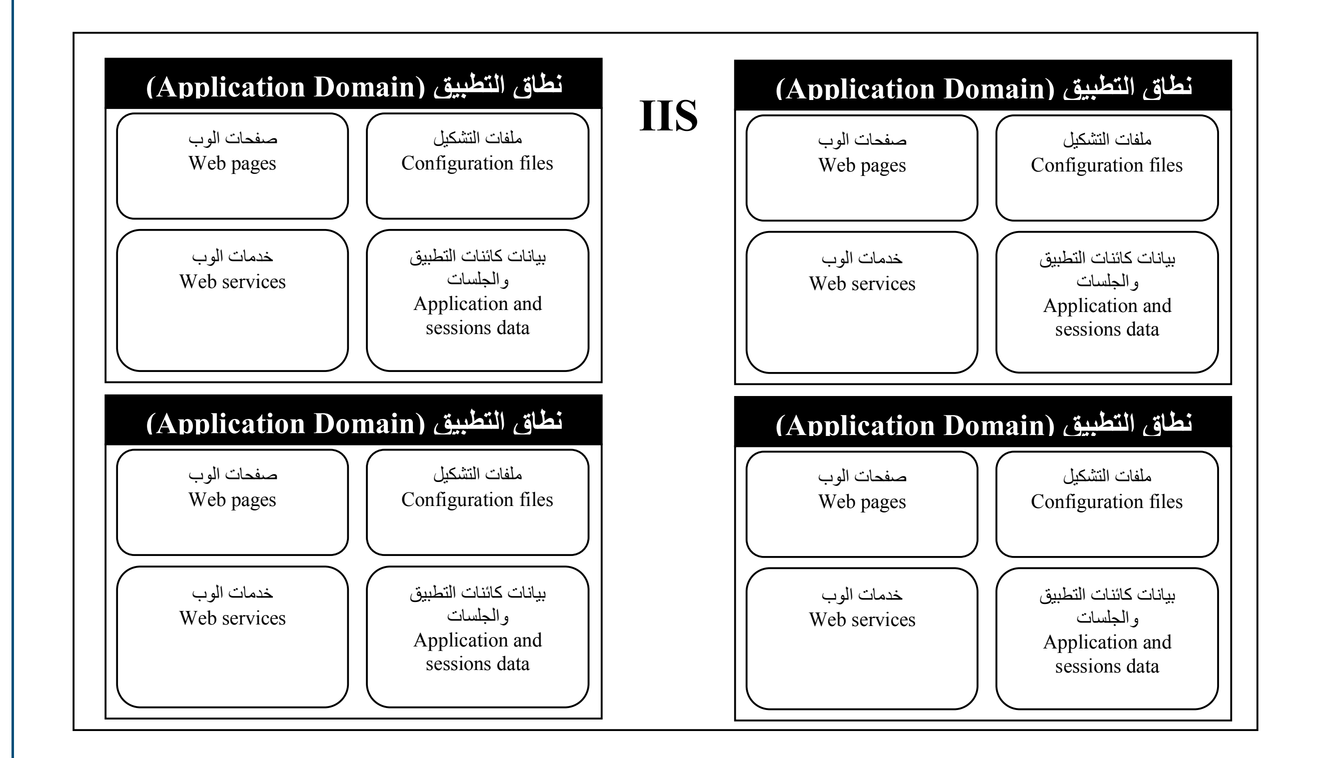
1. مفهوم تطبيقات وب
2. بنية المجلدات في مشروع ASP.NET
3. أنواع الملفات الأساسية التي تتعامل معها ASP.NET
4. عناصر تحكم HTML من طرف المخدم
5. أحداث عناصر تحكم HTML من جهة المخدم
6. الصفوف القاعدية لعناصر التحكم ، خواصها والأحداث التي تطلقها
7. ملفات global.asax و أحداث التطبيق
8. الإعدادات في تطبيقات ASP.NET وملفات machine.config، web.config

مفهوم تطبيقات الوب

يصعب في بعض الأحيان تحديد ماهية تطبيق الوب و حدوده بعكس ما هو في التطبيقات التقليدية (التي يمكن إطلاقها بتشغيل ملف تنفيذي مستقل) .

تتكون تطبيقات الوب من مجموعة من صفحات الوب. نتيجة لهذا التقسيم إلى صفحات، يمكن للمستخدم أن يدخل إلى التطبيق من عدة صفحات، كما يمكنه أن ينتقل من صفحة إلى أخرى من خلال الضغط على وصلة (Hyperlink). كما يمكنه في بعض الأحيان، أن ينتقل من مخدم وب إلى مخدم آخر من خلال الضغط على وصلة. لذا يكون السؤال المطروح عادةً: ما هي حدود تطبيقات الوب.

تشترك جميع الصفحات في تطبيق ASP.NET بمجموعة من الموارد (Resources) وبمجموعة من الإعدادات (Configurations). تكون مجموعة الموارد والإعدادات مخصصة لتطبيق دون آخر حتى ولو كان التطبيق الآخر موجود على نفس المخدم. إذ يتم عزل كل تطبيق ضمن ما يسمى نطاق التطبيق (Application Domain) ويكون هذا العزل على مستوى الذاكرة، والمعالجة، ومعلومات الجلسات، والذاكرة الخبيئة. نتيجةً لذلك، يتم تعريف نطاق تطبيق الوب بأنه مجموعة الصفحات و الوحدات والمقابض (handlers) التي يمكن الوصول إليها من خلال مجلد افتراضي على مخدم الوب (أو من خلال المجلدات الفرعية الموجودة ضمن المجلد الأفتراضي الأساسي وتبعاً للسماحيات المختارة).



أنواع الملفات في ASP.NET:

اللاحقة	الوصف
.aspx	هذه اللاحقة خاصة بصفحات الوب في ASP.NET. تحتوي تعبيرات لغة التأشير التي تحدد تصميم واجهة المستخدم. قد تحتوي هذه الصفحات رماز التطبيق الذي يتم حصره ضمن وسم <SCRIPT runat="server"></SCRIPT>
.ascx	تستخدم هذه اللاحقة مع ملفات عناصر التحكم الخاصة بالمستخدم . هذه الصفحات مشابهة لسابقاتها إلا فيما يتعلق بإمكانية الوصول إليها مباشرة. لاستخدام هذه العناصر لا بد من استضافتها ضمن صفحة أخرى . تمكن هذه العناصر من تصميم قطع من واجهة المستخدم قابلة لإعادة الاستخدام.
.asmx	تمثل الملفات التي تستخدم هذه اللاحقة مجموعات الطرق الخاصة بخدمات الوب. يمكن استدعاء هذه الطرق عبر الانترنت . تعمل خدمات الوب بشكل مختلف عن صفحات الوب لكنها تشترك معها في الموارد و الإعدادات (configuration) و الذاكرة الخاصة بالتطبيق.
Web.config	يستخدم ملف web.config تنسيق XML . تم تخصيص هذا الملف في ASP.NET لإعداد التطبيق. يتضمن هذا الملف إعدادات خاصة بأمن التطبيق ، إدارة الحالة ، إدارة الذاكرة و غير ذلك.

Global.asax	هذا الملف هو الملف العام للتطبيق. يتم استخدام هذا الملف لتعريف المتحولات العامة ، حيث أن هذه المتحولات تكون متاحة لأي صفحة ضمن تطبيق الوب كما يمكن استخدام هذا الملف لتعريف طرق مقابض الأحداث (Event Handler) العامة كالطريقة الخاصة بمقبض حدث بدء تشغيل التطبيق.
.cs	تستخدم هذه اللاحقة مع الملفات الحاوية على الرماز في الخلفية (background code) . تمكن هذه الملفات في ASP.NET من فصل منطق التطبيق عن واجهة المستخدم في صفحة الوب.

ما يمكن لتطبيق ASP.NET أن يحوي أنواع الملفات المعروفة مثل CSS. أو HTML. أو أنواع ملفات الصور مثل JPG. أو GIF.

المجلدات الأساسية ضمن تطبيق ASP.NET:

عادة ما يتم التركيز على مجموعة من الممارسات التي تساعد على بناء تطبيق وب ناجح . من أهم هذه الممارسات مراعاة استخدام هيكلية مجلدات مناسبة ضمن التطبيق .

الجدول التالي يوضح بعض هيكلية و عمل المجلدات التي تقوم بيئة تطوير Visual studio بإنشاءها ضمن مجلد التطبيق حسب الحاجة:

المجلد	الوصف
Bin	يحتوي هذا المجلد على جميع مكونات NET. المترجمة (مكتبات الربط الديناميكي DLL) التي يتم استخدامها من قبل التطبيق إضافة إلى المكونات المخصصة التي يضيفها المستخدم . تقوم ASP.NET باكتشاف المكونات االمجمعة و تمكن أي صفحة وب ضمن التطبيق من استخدامها . توفر هذه الطريقة سهولة كبيرة مقارنة مع العمل مع مكونات COM التي تستلزم تسجيل المكون قبل استخدامه.
App_Code	يحتوي هذا المجلد الرماز المصدري الذي تتم ترجمته ديناميكياً لاستخدامه ضمن التطبيق (يحتوي هذا المجلد الرماز المصدري و ليس الملفات المجمعة كما هو الحال في Bin)
App_GlobalResources	تستخدم لتخزين الموارد العامة المتاحة لجميع الصفحات ضمن التطبيق .
App_LocalResources	يشبه هذا المجلد المجلد السابق بفرق كون الملفات متاحة لصفحة محددة
App_WebReferences	يستخدم هذا المجلد لتخزين مراجع لخدمات الوب المستعملة من قبل التطبيق.
App_Data	يستخدم هذا المجلد لتخزين البيانات بما يتضمن قواعد بيانات SQL Express 2005 و ملفات XML
App_Themes	يستخدم لتخزين السمات التي يستخدمها التطبيق.

مقدمة إلى عناصر التحكم من جهة المخدم:

طرحت ASP.NET مفهوماً جديداً في إنشاء صفحات الوب. يستخدم هذا النموذج عناصر التحكم من جهة المخدم.
يتم إنشاء هذه العناصر و تشكيلها كأغراض ، تعمل هذه الأغراض على المخدم .
تقوم هذه الأغراض بتوليد خرج HTML الخاص بها.
تعمل هذه العناصر بشكل مشابه لمثيلاتها في تطبيقات Windows العادية من حيث قدرتها على الحفاظ على الحالة و إطلاق الأحداث التي يمكن معالجتها ضمن الرماز.
توفر ASP.NET مجموعتين أساسيتين من عناصر التحكم من جهة المخدم:

- عناصر تحكم HTML من جهة المخدم: و هي المقابلات لتأشيرات HTML القياسية و لكن من جهة المخدم. تعتبر عناصر التحكم هذه مثالية لمبرمجي الوب الذين يفضلون العمل على تأشيرات HTML المألوفة لديهم. كما أن عناصر التحكم هذه مناسبة لأولئك الذين يريدون الانتقال من تطوير صفحات ASP إلى تطوير صفحات ASP.NET كونها لا تتطلب قدر كبير من التغيير عما تعوده المطورون الذين تعودوا العمل باستخدام ASP .

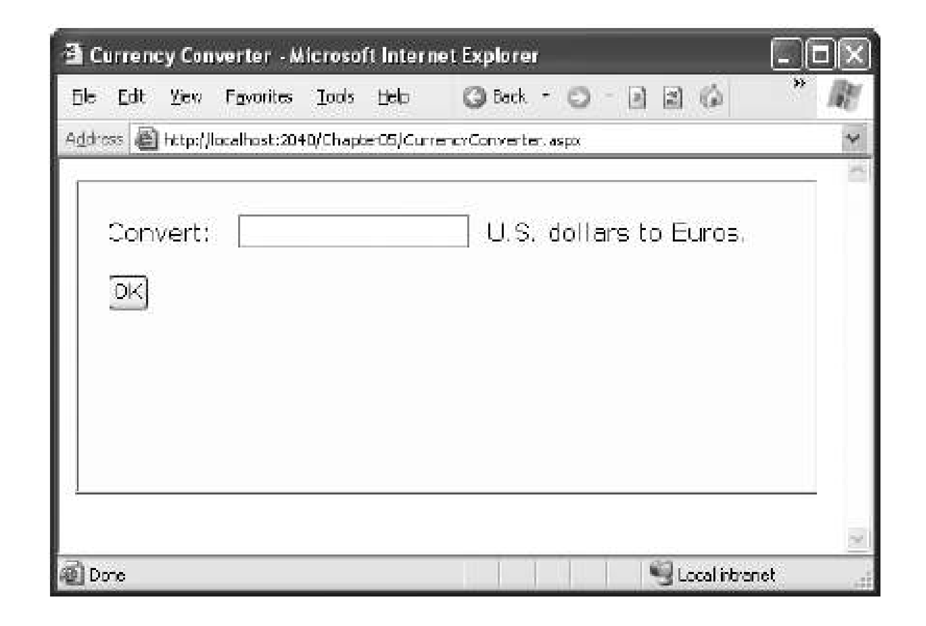
  لعناصر تحكم HTML من جهة المخدم ثلاث ميزات أساسية:
  1. تقوم هذه العناصر بتوليد واجهتها الخاصة
  2. تحتفظ هذه العناصر بمعلومات الحالة
  3. تقوم عناصر التحكم هذه بإطلاق الأحداث من جهة المخدم

- عناصر تحكم الوب: تشابه هذه العناصر عناصر HTML من جهة المخدم بشكل كبير ولكنها توفر:
  1. نموذج غرضي أكثر غنى، من حيث تنوع الخصائص و تفاصيل التنسيق و الأنماط
  2. أحداث أكثر
  3. بيئة تطوير أقرب إلى تلك المستخدمة في تطوير تطبيقات windows مثل عناصر GridView و Calendar و عناصر التحقق من الصحة.

عناصر تحكم HTML من جهة المخدم:

تحويل صفحة HTML إلى صفحة ASP.NET:

سنحاول في هذا الجزء من الجلسة استخدام مثال للوصول إلى فهم أكبر لعناصر تحكم HTML من جهة المخدم.
يُظهر الشكل رقم (Figure2) واجهة صفحة وب خاصة بتطبيق تحويل عملة من الدولار إلى مقابلتها باليورو.
التطبيق في هذه المرحلة هو مجرد واجهة ، تمت صياغتها بتأشيرات HTML قياسية و لن تتسبب عملية نقر الزر OK أية استجابة.



فيما يلي نص HTML القياسي لهذه الصفحة:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Currency Converter</title>
</head>
<body>
<form method="post">
<div>
Convert: 
<input type="text" />
 U.S. dollars to Euros.
<br /><br />
<input type="submit" value="OK" />
</div>
</form>
</body>
</html>
```

إن أسهل طريقة لتحويل هذه الصفحة إلى صفحة ASP.NET هي بتوليد نموذج وب جديد باختيار:

1. Websites > Add New Item
2. ثم اختيار Web Form و تحديد اسم للصفحة مثلاً CurrencyConverter.aspx
3. تأكد من تفعيل خيار فصل الرماز في ملف مستقل ثم اضغط Add لإنشاء الصفحة.
4. قم بحذف جميع محتويات الصفحة عدا موجه <%@page%> .
5. يزود هذا الموجه ASP.NET بالمعلومات الأساسية عن كيفية ترجمة الصفحة ، اللغة المستخدمة و طريقة الوصول إلى مقابض الأحداث و موقع ملف الرماز العامل في خلفية الصفحة.
6. قم بنسخ محتوى ملف HTML إلى الصفحة الجديدة مباشرة بعد تأشيرة الموجه Page .

ما يزال النموذج حتى هذه المرحلة غير عامل ، ولا بد لنا لجعله يعمل من إضافة الواصفة "runat="server إلى جميع التأشيرات التي نرغب بتحويلها إلى عناصر تحكم من جهة المخدم، كما لابد من إضافة الواصفة ID إلى جميع هذه العناصر.
يمكن التفاعل مع عناصر التحكم باستخدام الاسم الفريد الذي تم إسناده إلى الواصفة ID و الذي سيتم استخدامه كمرجع إلى عناصر التحكم تلك.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CurrencyConverter.aspx.cs" Inherits="CurrencyConverter" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Currency Converter</title>
</head>
<body>
<form runat="server">
<div>
Convert:  
<input type="text" ID="US" runat="server" />
  U.S. dollars to Euros.
<br /><br />
<input type="submit" value="OK" ID="Convert" runat="server"
OnServerClick="Convert_ServerClick" />
<br /><br />
<div style="font-weight: bold" ID="Result" runat="server"></div>
</div>
</form>
</body>
</html>
```

صفوف عناصر تحكم HTML من طرف المخدم:

اسم الصف	عنصر HTML	
HtmlTextArea	<textarea>	علبة نصية مخصصة لإدخال نص متعدد الأسطر.
HtmlInputImage	<input type="image">	مشابه لعنصر <img> ولكن يمكّن إدراج صورة يمكن النقر عليها لإرسال النموذج.
HtmlInputFile	<input type="file">	علبة نصية و زر مخصص لعملية اختيار ملف لتحميله على مخدم الوب.
HtmlInputHidden	<input type="hidden">	يقوم باحتواء معلومات نصية ليتم إرسالها دون أن تكون مرئية على النموذج.
HtmlSelect	<select>	قائمة منسدلة أو قائمة عادية تحتوي

مجموعة عناصر و تمكّن اختيار عنصر أو أكثر من المجموعة.		
تمثل معلومات الترويسة التي تتضمن بيانات عن الصفحة . معظم هذه البيانات لا يتم إظهارها. هذه العناصر هي الوحيدة التي لا يتم وضعها ضمن تأشيرة عنصر التحكم <form>	<head> and <title>	HtmlHead and HtmlTitle
عنصر التحكم هذا يمثل مجموعة من عناصر HTML التي ليس لها صفوف خاصة بها فمثلاُ إذا قمنا بإضافة الواصفة runat="server" إلى تأشيرة <div> سيتم استخدام الصف HtmlGenericControl للتعامل مع عنصر تحكم HTML الناتج.	Any other HTML element.	HtmlGenericControl

يمتلك كل من هذه الصفوف مجموعة من الخصائص من أهمها التالية:

الخصائص الهامة	عنصر التحكم
HRef, Name, Target, Title	HtmlAnchor
Src, Alt, Align, Border, Width, Height	HtmlImage
Checked	HtmlInputCheckBox and HtmlInputRadioButton
HtmlInputText	Value
HtmlTextArea	Value
HtmlInputImage	Src, Alt, Align, Border
HtmlSelect	Items (collection)
HtmlGenericControl	InnerText and InnerHtml

إضافة الرماز الخاص بتـــطبيق تحويل العملات:

لبث الحياة في تطبيقنا لا بد من إضافة عبارات رماز ASP.NET إلى مقابض الأحداث (Event Handlers) لأن نماذج الوب في ASP.NET موجهة بالأحداث بمعنى أن كل جزء من الرماز يتم تنفيذه تجاوباً مع حدث معين.
الحدث الأهم في مثالنا هو الضغط على زر (Submit) ، حيث يوفر عنصر التحكم HTMLInputButton إمكانية الاستجابة لحدث النقر على الزر SeverClick من طرف المخدم.
يتم استخدام التأشيرة <div> لإدراج نتيجة التحويل كما هو موضح ضمن الرماز التالي:

```
<%@ Page Language="C#" AutoEventWireup="true"
```

```
CodeFile="CurrencyConverter.aspx.cs" Inherits="CurrencyConverter" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Currency Converter</title>
</head>
<body>
<form runat="server">
<div>
Convert:  
<input type="text" ID="US" runat="server" />
  U.S. dollars to Euros.
<br /><br />
<input type="submit" value="OK" ID="Convert" runat="server"
OnServerClick="Convert_ServerClick" />
<br /><br />
<div style="font-weight: bold" ID="Result" runat="server"></div>
</div>
</form>
</body>
</html>
```

أما الرماز العامل في الخلفية فيتم تخزينه ضمن الملف *CurrencyConverter.aspx.cs*:

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
public partial class CurrencyConverter : System.Web.UI.Page
{
protected void Convert_ServerClick(object sender, EventArgs e)
{
decimal USAmount = Decimal.Parse(US.Value);
decimal euroAmount = USAmount * 0.85M;
Result.InnerText = USAmount.ToString() + " U.S. dollars = ";
Result.InnerText += euroAmount.ToString() + " Euros.";
}
}
```

يبدأ ملف الرماز العامل في الخلفية،كما نلاحظ، بمجموعة من التصريحات الخاصة بتأمين الوصول إلى فضاءات الأسماء التي يحتاجها التطبيق.

يتم تعريف صف الصفحة بالكلمة المفتاحية Partial لأن رماز الصف هو عبارة عن دمج للرماز الذي قمنا بكتابته مع رماز إضافي (مخفي) تولده ASP.NET أوتوماتيكياً.

يُعرف الرماز الذي تولده ASP.NET جميع عناصر التحكم التي يتم استخدامها ضمن الصفحة مما يمكن الوصول لهذه العناصر بشكل مباشر من خلال اسمها.

يتم في مثالنا تعريف مقبض أحداث وحيد، يقوم هذا المقبض باسترجاع القيمة من العلبة النصية ، تحويلها إلى قيمة رقمية ثم ضربها بنسبة تحويل محددة.

نلاحظ أن مقبض الحدث يقبل قيمتين لغرضين هما الغرض Sender و الغرض e.

تسمح هذه الأغراض للرماز بالتعرف على عنصر التحكم الذي أرسل الحدث (عبر الغرض Sender) و الاستحصال على جميع المعلومات المرتبطة بالحدث (عبر الغرض e).

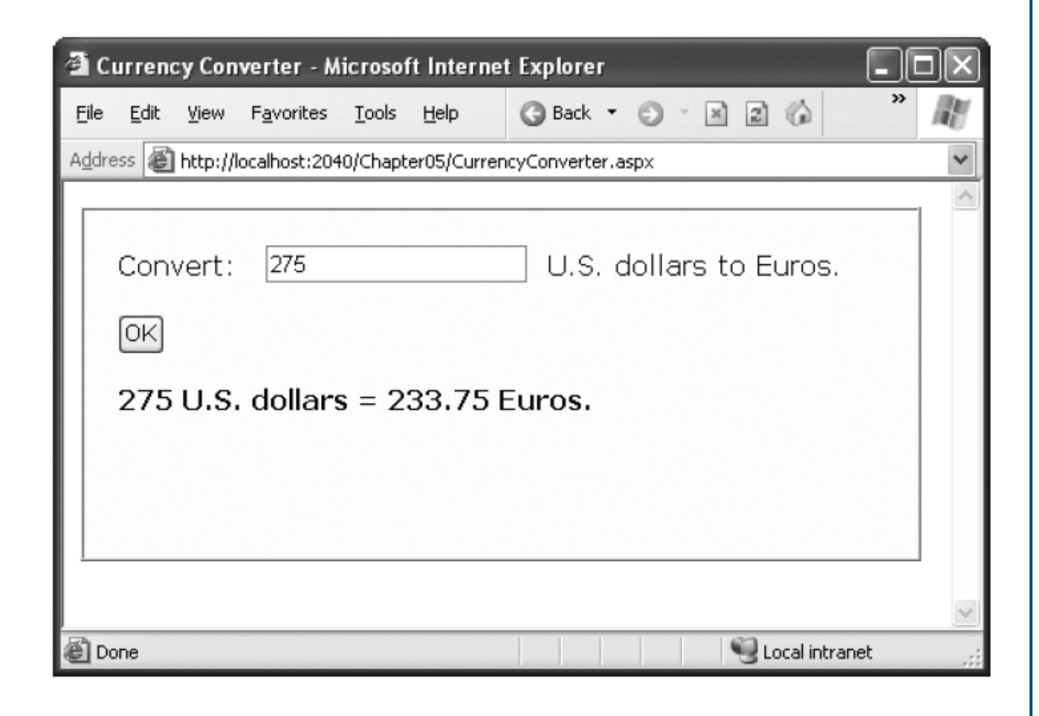
يتم تحديد اسم مقبض الحدث لعنصر التحكم عبر الواصفة OnServerClick ضمن التأشيرة <input> الخاصة بالزر.

ملاحظة: تختلف عناصر تحكم HTML من جهة المخدم عن عناصر تحكم الوب .حيث لا توفر الأولى إمكانية إنشاء مقابض الأحداث باستخدام نافذة Properties في بيئة التطوير بل يجب كتابتها يدوياً كذلك هي الحال بالنسبة لتعديل قيمة الواصفة OnServerClick بما يوافق اسم المقبض المستخدم.

معالجة الأحداث:

عندما يقوم المستخدم بالنقر على زر Convert سيتم إرسال الصفحة إلى مخدم الوب عندها تحدد ASP.NET الرماز الواجب تشغيله من خلال قيمة الواصفة OnServerClick

```
<input type="submit" value="OK" ID="Convert"
OnServerClick="Convert_ServerClick" runat="server">
```



تستخدم ASP.NET الصيغة السابقة لربط أي مقبض حدث خاص بتأشيرة عنصر التحكم ،حيث يتم استعمال اسم الحدث مسبوقاً بالكلمة On.

إذا أردنا مثلاً معالجة الحدث المسمى ServerChange نقوم باسناد قيمة تعبر عن اسم الطريقة التي ستقوم بمعالجة الحدث إلى الواصفة OnServerChange ضمن تأشيرة عنصر التحكم.

ملاحظة: يمكننا هنا أن نفهم سبب الخطأ متكرر الحدوث المتمثل بظهور رسالة خطأ عندما نقوم بحذف نص الرماز المولد آلياً والخاص

بمقبض الحدث الناتج عن النقر المزدوج على عنصر التحكم.
ينتج هذا الخطأ عن توليد واصفة ضمن تأشيرة العنصر الذي تم النقر المزدوج و ربط الحدث بنص مقبض الحدث .
لحل المشكلة يكفي إزالة هذه الواصفة من تأشيرة عنصر التحكم.

ملاحظة: لا يتطلب غرض الصفحة في ASP.NET عملية الربط بالأحداث.
يتم في غرض الصفحة ربط الأحداث ذات الأسماء المحددة مسبقاً تلقائياً في حال استخدمت الاسم الصحيح و عدد المعاملات الصحيح.
تسمى هذه الخاصة الربط الآلي للأحداث(Automatic event wireup).

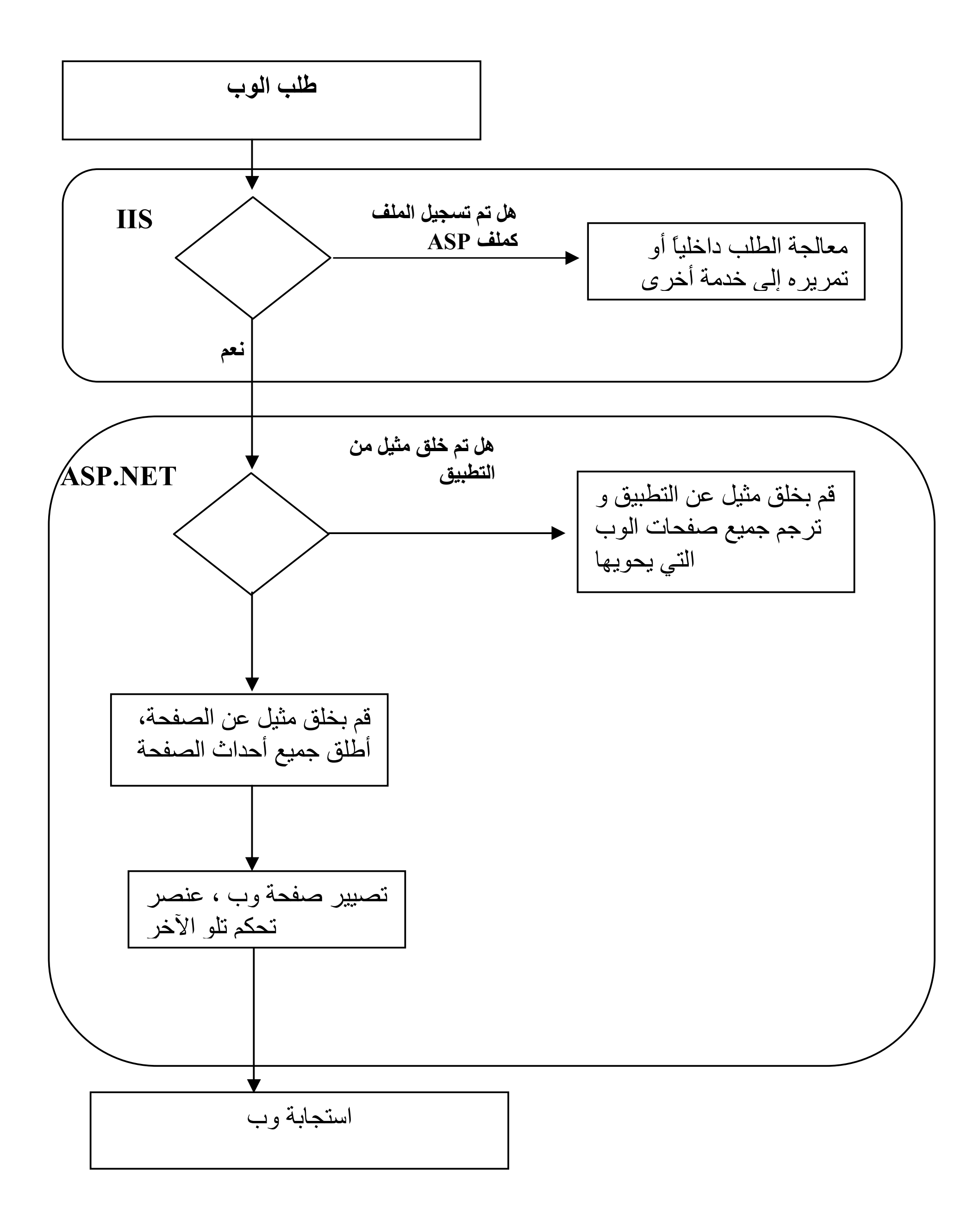
تسمح ASP.NET باستخدام تقنية أخرى لربط الأحداث و ذلك عبر النص البرمجي إذ يمكننا مثلاً إضافة النص البرمجي التالي إلى مقبض الحدث PageLoad():

```
Convert.ServerClick += this.Convert_ServerClick;
```

ملاحظة: قد لا يكون ربط الأحداث يدوياً مهماً إلا في بعض الحالات .من أهم هذه الحالة حالة إضافة عناصر تحكم بشكل ديناميكي في زمن التشغيل إلى الصفحة.
ما الذي يحدث خلف الكواليس في مثال محول العملة:

1. يتم إرسال طلب الصفحة إلى مخدم الوب (يكون هذا المخدم غالباً ISS في حال العمل ضمن بيئة استثمار حقيقية).
2. يقوم مخدم الوب بتحديد كون اللاحقة ASPX مسجلة ليتم معالجتها عبر ASP.NET.
3. يتم تمرير الطلب إلى الإجراء العامل من ASP.NET (لن تتدخل ASP.NET في حال كانت اللاحقة هي asp. أوملفات .html)
4. عند استدعاء صفحة ما من هذا التطبيق لأول مرة تقوم ASP.NET بخلق نطاق للتطبيق و تترجم رماز صفحة الوب لضمان أداء أمثلي كما تقوم بتخزين الملفات المترجمة ضمن المجلد:
c:\Windows\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files.
5. في حال استدعاء أية صفحة تمت ترجمتها مسبقاً تقوم ASP.NET بإعادة استخدام النسخة المترجمة من الصفحة.
6. عندما انتهاء تنفيذ الرماز تقوم ASP.NET بالطلب من كل عنصر تحكم أن يقوم بتصيير نفسه (إضافة تأشيرات إلى الخرج إلى تأشيرات HTML المناسبة.
ملاحظة : تضيف ASP.NET أحياناً ،أثناء تصيير عناصر التحكم، نصوص لغة خطاطية من جهة الزبون مثل JavaScript أو DHTML إذا تحققت من دعم المستعرض لها.
7. يتم إرسال الصفحة المتشكلة (كناتج خرج لتصيير عناصر التحكم و خرج ASP.NET) إلى الزبون و يتم إنهاء التطبيق.

فيما يلي مخطط يوضح المراحل المختلفة لتنفيذ طلب صفحة وب.



ملاحظة:يعمل تطبيق ASP.NET ضمن بيئة معزولة على المخدم ،حيث يستطيع الزبون رؤية النتائج فقط عندما تنتهي عملية معالجة الصفحة ويتم تحرير غرض الصفحة من الذاكرة.

معالجة الأخطاء:

يتوقع التطبيق، في مثالنا (محول العملة) ، من المستخدم إدخال رقم قبل ضغط زر Convert. ولكن ماذا لو ضغط المستخدم زر Convert قبل إدخال أي رقم؟

سيتم إرسال سلسلة نصية أو مجموعة من المحارف الخاصة التي لن يتمكن التطبيق من تحويلها بنجاح إلى رقم ، مما سيولد خطأ عند محاولة استخدام التطبيق للطريقة Decimal.Parse() وسيتوقف عمل التطبيق، بعدها سيتم إرسال صفحة الخطأ إلى المستخدم مع وصف للخطأ الحاصل .

يمكن إزالة الخطأ في مثالنا باستخدام الطريقة Decimal.TryParse() عوضاً عن Decimal.Parse().

توفر الطريقة Decimal.TryParse() معاملين الأول خاص بالقيمة المراد تحويلها و الثاني هو معامل خرج يحتوي النتيجة بعد التحويل ، أما القيمة المعادة من هذه الطريقة فهي قيمة منطقية (True , False).

ملاحظة: سنتعرف في جلسة لاحقة على كيفية معالجة الأخطاء باستخدام بنى أكثر عمومية

تحسين تطبيق محول العملات:

سنقوم في هذا الجزء من الجلسة بتحسين تطبيقنا عبر وظائف إضافية و استخدام عناصر تحكم مختلفة.

إضافة عملات متعددة:

لتمكين التعامل مع عدة عملات سنقوم باستخدام قائمة منسدلة.

يمكننا إضافة عنصر تحكم HTML خاص بالقائمة المنسدلة باستخدام التأشيرة <Select> مع مراعاة عدم إغفال اسناد قيمة للواصفة runat="server".

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CurrencyConverter.aspx.cs" Inherits="CurrencyConverter" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Currency Converter</title>
</head>
<body>
<form runat="server">
<div>
Convert:  
<input type="text" ID="US" runat="server" />
  U.S. dollars to  
<select ID="Currency" runat="server" />
<br /><br />
<input type="submit" value="OK" ID="Convert"
OnServerClick="Convert_ServerClick" runat="server" />
<br /><br />
<div style="font-weight: bold" ID="Result" runat="server"></div>
</div>
</form>
</body>
```

```
</html>
```

بعد أن قمنا بإنشاء عنصر تحكم قائمة منسدلة فارغ، سنستخدم الرماز لإضافة عناصر إليه.

```
protected void Page_Load(Object sender, EventArgs e)
{
if (this.IsPostBack == false)
{
Currency.Items.Add("Euro");
Currency.Items.Add("Japanese Yen");
Currency.Items.Add("Canadian Dollar");
}
}
```

تم استخدام الخاصة Items. تسمح هذه الخاصة بإضافة و حذف عناصر من القائمة.

قبل إضافة أي عنصر إلى القائمة يجب التأكد بأن الرماز لن ينفّذ إلا عند أول مرة يتم فيها إرسال الصفحة للمستخدم ، لكي لا ننتهي إلى سيناريو يقوم بإضافة عناصر بشكل متكرر إلى القائمة في كل مرة يتم فيها الدخول إلى الصفحة.

يتم ،لغرض التأكد من هذه النقطة، التحقق من الخاصة IsPostBack للصفحة الحالية.

في حال كون قيمة هذه الخاصة الموروثة من صف Page الرئيسي مساوية لـــ false فمعناه أن تلك هي المرة الأولى التي يتم فيها إنشاء تلك الصفحة لهذا المستخدم و من الآمن تأهيلها.

تخزين المعلومات ضمن القائمة:

يمكن وكما نعلم من التعامل مع HTML أن يكون كل خيار ضمن القائمة المنسدلة مرتبط بالنص الذي يظهر للمستخدم.

كذلك يمكن اسناد قيمة غير ظاهرة مرتبطة بهذا الخيار ، يعبر عنها في HTML بالواصفة Value.

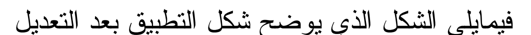
للتحكم بالقيم المرتبطة بالخيارات نستخدام الغرض ListItem حيث يوفر هذا الغرض مشيد يسمح لنا بتحديد النص مع القيمة المرتبطة عند إنشائه.

```
protected void Page_Load(Object sender, EventArgs e)
{
if (this.IsPostBack == false)
{
// The HtmlSelect control accepts text or ListItem objects.
Currency.Items.Add(new ListItem("Euros", "0.85"));
Currency.Items.Add(new ListItem("Japanese Yen", "110.33"));
Currency.Items.Add(new ListItem("Canadian Dollars", "1.2"));
}
}
```

لاتمام عملنا يجب إعادة كتابة النص البرمجي لأخذ نوع العملة المختارة بعين الاعتبار:

```
protected void Convert_ServerClick(object sender, EventArgs e)
{
```

```
decimal amount = Decimal.Parse(US.Value);
// Retrieve the selected ListItem object by its index number.
ListItem item = Currency.Items[Currency.SelectedIndex];
decimal newAmount = amount * Decimal.Parse(item.Value);
Result.InnerText = amount.ToString() + " U.S. dollars = ";
Result.InnerText += newAmount.ToString() + " " + item.Text;
}
```


فيمايلي الشكل الذي يوضح شكل التطبيق بعد التعديل



قمنا في هذا التطبيق بتخزين معدل التحويل ضمن الواصفة Value. ولكن في تطبيقات أعقد قد يكون من الأفضل تخزين المعلومات ضمن قاعدة البيانات أو ضمن الذاكرة الخبيئة و تخزين قيمة المفتاح المعرف ضمن الواصفة Value.

إضافة الصور المرتبطة:

أصبح إدراج وظائف إضافية في تطبيقنا سهلاً، بسهولة إضافة زر ، فيمكننا مثلاً إضافة صورة تعبر عن منحني تغير قيمة التحويل كما يظهر في الرماز:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CurrencyConverter.aspx.cs" Inherits="CurrencyConverter" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Currency Converter</title>
</head>
<body>
<form runat="server">
<div>
Convert:  
<input type="text" ID="US" runat="server" />
  U.S. dollars to  
<select ID="Currency" runat="server" />
```

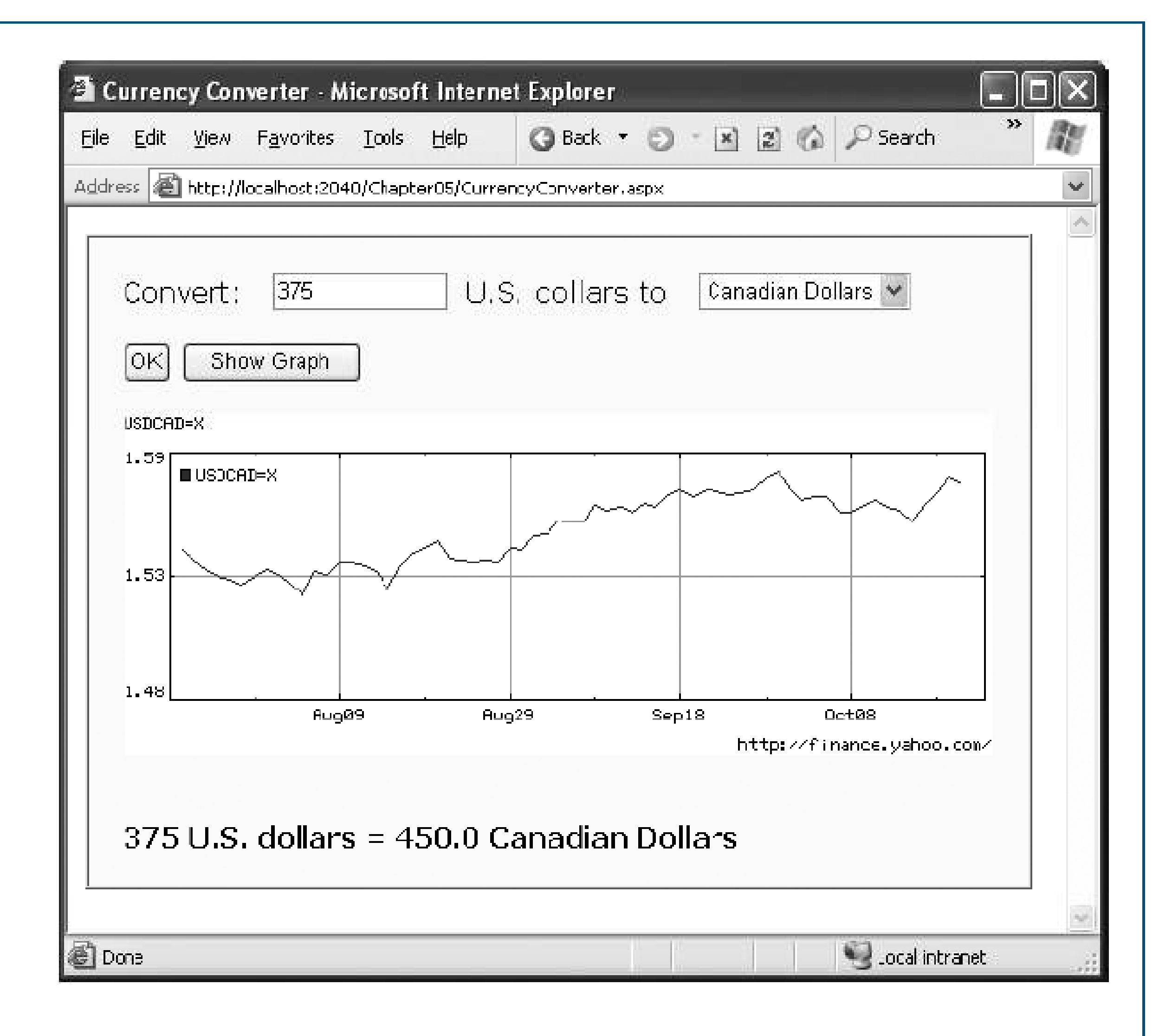
```
<br /><br />
<input type="submit" value="OK" ID="Convert"
OnServerClick="Convert_ServerClick" runat="server" />
<input type="submit" value="Show Graph" ID="ShowGraph" runat="server" />
<br /><br />
<img ID="Graph" scr="" alt="Currency Graph" runat="server" />
<br /><br />
<div style="font-weight: bold" ID="Result" runat="server"></div>
</div>
</form>
</body>
</html>
```

يكون عنصر تحكم الصورة الذي قمنا بإضافته فارغاً في مرحلة ما قبل الاختيار لذلك من المنطقي إخفاؤه.

```
protected void Page_Load(Object sender, EventArgs e)
{
if (this.IsPostBack == false)
{
Currency.Items.Add(new ListItem("Euros", "0.85"));
Currency.Items.Add(new ListItem("Japanese Yen", "110.33"));
Currency.Items.Add(new ListItem("Canadian Dollars", "1.2"));
}
Graph.Visible = false;
}
```

أما الرماز الخاص بزر إظهار الصورة فهو التالي:

```
protected void ShowGraph_ServerClick(Object sender, EventArgs e)
{
Graph.Src = "Pic" + Currency.SelectedIndex.ToString() + ".png";
Graph.Visible = true;
}
```



إعداد الأنماط:

إضافة إلى المجموعة المحدودة من الخصائص التي توفرها عناصر HTML يمكننا الوصول إلى واصفات CSS.

تتم عملية التحكم بقيم واصفات CSS من خلال استخدام المجموعة Style.

يتم تحديد اسم الواصفة و القيمة المراد اسنادها إلى تلك الواصفة كما يلي:

```
ControlName.Style["AttributeName"] = "AttributeValue";
```

يمكننا استخدام هذا التعبير في مثالنا (محول العملات) لتمييز الإدخال الخاطئ بنمط خاص.

يجب ألا ننسى أنه لا بد من إعادة قيمة اللون إلى القيمة الأصلية من أجل الإدخال الصحيح لأن عناصر التحكم تستخدم View State لحفظ جميع إعدادات العنصر بما فيها قيم خصائص النمط.

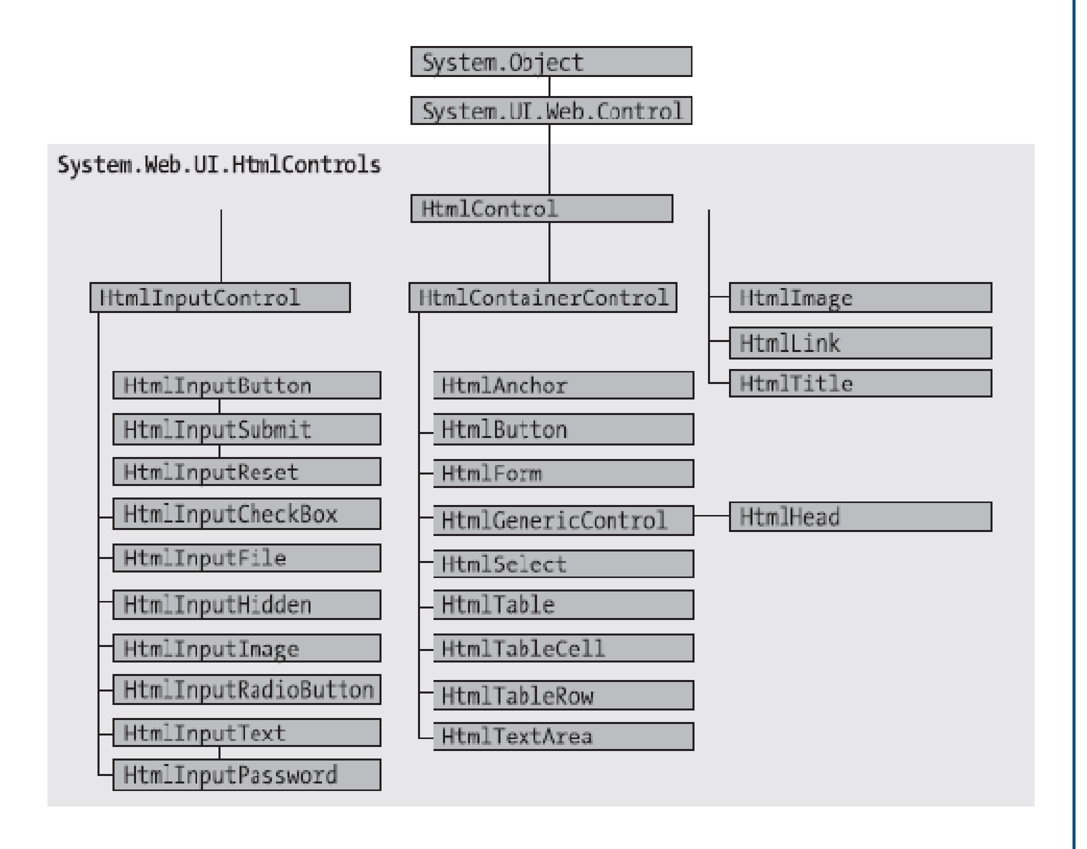
```
protected void Convert_ServerClick(object sender, EventArgs e)
{
decimal amount = Decimal.Parse(US.Value);
if (amount <= 0)
```

```
{
Result.Style["color"] = "Red";
Result.InnerText = "Specify a positive number";
}
else
{
Result.Style["color"] = "Black";
// Retrieve the selected ListItem object by its index number.
ListItem item = Currency.Items[Currency.SelectedIndex];
decimal newAmount = amount * Decimal.Parse(item.Value);
Result.InnerText = amount.ToString() + " U.S. dollars = ";
Result.InnerText += newAmount.ToString() + " " + item.Text;
}
}
```

قمنا في هذا الجزء من الجلسة بإتمام تطبيق محول العملات وتعرفنا من خلاله على كيفية إجراء حسابات بسيطة، التعامل مع الصور المرتبطة و التنسيق الديناميكي.

صفوف عناصر تحكم HTML:

تستخدم صفوف NET. الوراثة لتأمين عملية مشاركة الوظائف.
على سبيل المثال يرث كل عنصر تحكم HTML من الصف القاعدي HtmlControl، يوفر هذا الصف وظائف أساسية لجميع عناصر تحكم HTML من جهة المخدم.



صفوف ASP.NET المستخدمة لعناصر تحكم HTML من جهة المخدم

الجزء التالي من الجلسة سيقوم بتفصيل صفوف ASP.NET المستخدمة لعناصر تحكم HTML من جهة المخدم.

توفر عناصر تحكم HTML من جهة المخدم عادة خصائص تطابق بشكل كبير واصفات التأشيرات.
على سبيل المثال يوفر صف HtmlImage خصائص (Align, Border, Src, Height ,Width). لهذا السبب يجد المطورون الذين اعتادوا التعامل صفحات Html سهولة في التعامل مع هذا النمط من عناصر التحكم.

أحداث عناصر تحكم HTML:
توفر عناصر تحكم HTML أيضاً إمكانية الاستجابة لأحد حدثين الأول ServerClick و الثاني ServerChange.

1. ServerClick: يمكّن هذا الحدث من معالجة عملية النقر على العنصر من طرف المخدم. توفر هذا الحدث معظم العناصر مثل عناصر الأزرار و عناصر HtmlAnchor.
2. ServerChange: يتم إطلاق هذا الحدث عندما يتم تعديل النص أو الاختيار المحدد في عنصر التحكم.

ملاحظة: في حالة HtmlAnchor يمكن لعنصر التحكم هذا العمل بطريقتين:

1. ارتباط مباشر، حيث يقوم بإعادة توجيه التطبيق إلى صفحة معينة
2. إرسال الصفحة عودياً و عندها سيتم تشغيل النص البرمجي ضمن مقبض الحدث

الحدث	عنصر التحكم الذي يوفر هذا الحدث
ServerClick	HtmlAnchor, HtmlButton, HtmlInputButton, HtmlInputImage, HtmlInputReset
ServerChange	HtmlInputText, HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden, HtmlSelect, HtmlTextArea

الأحداث المتقدمة المستعملة مع عنصر تحكم HtmlInputImage:

نعلم أن كل حدث يقوم بتمرير معاملين الأول يعرّف الغرض (في حالتنا عنصر التحكم) الذي قام بإطلاق الحدث والثاني هو عنصر خاص يتضمن معلومات إضافية عن الحدث.
في المثال الذي تم استعراضه في هذه الجلسة تم استخدام المعامل e لتمرير غرض System.EventArgs فارغ.
لا يحتوي هذا الغرض على أية معلومات إضافية لكنه يوضع فقط لغرض الالتزام بعدد المعاملات التي يستخدمها الحدث.
في الحقيقة عنصر وحيد فقط من عناصر تحكم HTML يقوم بإرسال معلومات إضافية و هو عنصر التحكم HtmlInputImage.
يقوم هذا العنصر بإرسال غرض ImageClickEventArgs الذي يوفر خاصيتي X,Y.

تمثل هاتان الخاصتان إحداثيات الموقع الذي تم النقر عليه ضمن عنصر الصورة.
يمكننا باستخدام هذه المعلومات الإضافية الاستغناء عن مجموعة عناصر تحكم button و image map وتعويضها بعنصر تحكم HtmlInputImage وحيد.
المثال التالي imageTest.aspx يعرض كيفية استخدام عنصر تحكم HtmlInputImage وتحديد احداثيات النقطة التي تم فيها نقر العنصر:




```
public partial class ImageTest : System.Web.UI.Page
{
protected void ImgButton_ServerClick(Object sender,
ImageClickEventArgs e)
{
Result.InnerText = "You clicked at (" + e.X.ToString() +
", " + e.Y.ToString() + "). ";
if ((e.Y < 100) && (e.Y > 20) && (e.X > 20) && (e.X < 275))
{
Result.InnerText += "You clicked on the button surface.";
}
else
{
Result.InnerText += "You clicked the button border.";
}
}
```

الصف القاعدي HtmlControl:

الخاصية	الوصف
Attributes	توفر مجموعة جميع الواصفات
Controls	تزود مجموعة جميع عناصر التحكم المحتواة ضمن عنصر التحكم الحالي
Disabled	عند اسناد القيمة false إلى هذه الواصفة لن يستطيع المستخدم التفاعل مع عنصر التحكم كما لن يطلق العنصر أي حدث.
EnableViewState	هذه الخاصة مسؤولة عن إدارة الحالة. عند اسناد القيمة false إلى هذه الخاصة لن يحتفظ عنصر التحكم بأي قيمة للخواص أو التنسيقات و سيتم إعادة قيم جميع الخواص إلى القيمة البدائية عند كل عملية إرسال الصفحة. في حال اسناد القيمة True إلى هذا الخاصة سيتم استخدام حقل إدخال مخفي لتخزين المعلومات الخاصة بعنصر التحكم. تتم هذه العملية لضمان المحافظة على التعديلات التي تمت في زمن التشغيل من خلال الرماز.
Page	توفر مرجع إلى غرض الصفحة الذي يحتوي عنصر التحكم
Parent	توفر مرجع إلى الغرض الأب الحاوي على عنصر التحكم. في حال تم وضع عنصر التحكم بشكل مباشر ضمن الصفحة ستعيد هذه الخاصية مرجع إلى غرض الصفحة.
Style	يوفر مجموعة من خصائص أنماط Css التي يمكن استخدامها لتنسيق عنصر التحكم.
TagName	يحدد اسم عنصر HTML الموافق لعنصر التحكم مثلاً (img,div,..)
Visible	إذا تم اسناد القيمة false إلى هذه الخاصة سيتم إخفاء العنصر و لن يتم تصييره ضمن الصفحة.

ملاحظة: يمكن اسناد القيم البدائية لهذه الخواص بتحديد هذه القيم ضمن مقبض حدث تحميل الصفحة Page_Load .

كما يمكن أسنادها مباشرة ضمن تأشيرة عنصر التحكم في صفحة aspx بإضافة واصفة بنفس الاسم تحديد قيمتها كما يلي:

```
<img ID="Graph" runat="server" Visible="false" ... />
```

الصف HtmlContainerControl:

ترث جميع عناصر التحكم التي تتضمن صيغة التأشيرة الخاصة بها تأشيرة إغلاق (مثل <div></div> أو <form></form>) الصف القاعدي HtmlContainerControl يقدم هذا الصف خاصتين إضافيتين هما:

الخاصة	الوصف
InnerHtml	تحدد أو تعيد نص Html المحصور بين فتح و إغلاق التأشيرة، يتم تصيير النص كنص Html أي يمكن إضافة تأشيرات ضمنية كـــ لتنسيق النص.
InnerText	تشابه هذه الخاصة سابقتها بفرق كونها تقوم بتحويل المحارف الخاصة (مثل فتح و إغلاق التأشيرة) إلى مقابلاتها مثلاً سيتم تحويل (<) إلى < أي سيظهر النص المحدد كقيمة لهذه الخاصية تماماً كما تم إدخاله.

الصف HtmlInputControl:

يقدم هذا الصف مجموعة من الخصائص التي يمكن استخدامها مع تأشيرة <input> إذ كما نعلم يمكن لهذه التأشيرة أن تمثل أكثر من عنصر تحكم بحسب القيمة المحددة للواصفة type. مثلاً <input type="text">

الخاصة	الوصف
Type	تحدد هذه الخاصة نوع عنصر تحكم الإدخال ، فمثلاً عنصر التحكم الذي <INPUT Type="file"> سيعيد القيمة file لهذه الخاصة.
Value	تعيد محتوى عنصر التحكم على شكل سلسة محرفية. يمكن استخدام هذه الخاصة في استعادة المعلومات التي تم إدخالها ضمن عنصر تحكم الإدخال (كما في مثال محول العملات)

صف الصفحة Page:

أي صفحة نقوم بإنشائها هي صف مخصص يرث من الصف System.Web.UI.Page وبعملية الوراثة تلك تكتسب صفوف الصفحات التي ننشؤها خصائص و طرق مختلفة منها ما له دور في عمليات الخبء، التحقق من الصحة و تصيير و إظهار العناصر. فيمايلي سرد لأسماء بعض هذه الخصائص وعملها:

الخاصية	الوصف
IsPostBack	تعيد هذه الخاصة قيمة منطقية. تأخذ هذه الخاصة القيمة false في حال كانت تلك هي المرة الأولى التي يتم فيها إرسال الصفحة.

EnableViewState	هذه الخاصة مسؤولة عن إدارة الحالة. عند اسناد القيمة false إلى هذه الخاصة لن يحتفظ عنصر التحكم بأي قيمة للخواص أو التنسيقات و سيتم إعادتها للقيمة البدائية عند كل عملية إرسال الصفحة. أما في حال اسناد القيمة إلى True إلى هذه الخاصة يقوم عنصر التحكم باستخدام حقل إدخال مخفي لتخزين المعلومات الخاصة به لضمان الاحتفاظ بالتعديلات التي تمت في زمن التشغيل من خلال الرماز .
Application	تعيد مجموعة تحوي المعلومات المشتركة بين كل المستخدمين للموقع .
Session	تعيد مجموعة تحوي معلومات خاصة بمستخدم واحد ، ليتم استخدامها عبر أكثر من صفحة . يمكن مثلاً استخدام هذا الغرض لتخزين معلومات سلة الشراء للمستخدم حين تتم عملية الشراء عبر أكثر من صفحة .
Cache	تعيد هذه الخاصة مجموعة تمكن من تخزين الأغراض التي تستغرق وقتاً ليتم إنشاؤها بغرض إعادة استخدامها في صفحات أخرى لزبائن آخرين و دون الحاجة إلى إعادة إنشائها.
Request	يعيد مرجع إلى غرض HttpRequest الحاوي على معلومات حول طلب وب الحالي ، يمكن الاستفادة من هذه المعلومات لمعرفة نوع المستعرض الذي يستخدمه الزبون. كما و يستخدم بشكل أساسي لنقل معلومات من صفحة إلى أخرى عبر الخاصة QueryString
Response	تعيد مرجع إلى غرض HttpResponse الممثل للاستجابة التي سترسلها Asp.NET إلى مستعرض الزبون. يمكن استخدام هذا الكائن أيضاً لإعادة توجيه المستخدم إلى صفحة أخرى أو عند العمل مع الكعكات.
Server	تعيد مرجع إلى غرض HttpServerUtitlity الذي يسمح بأداء مجموعة مختلفة من المهام. على سبيل المثال، تمكّن من تحويل النص إلى نمط آمن ليتم استخدامه ضمن URL أو في تأشيرة HTML
User	سوف تعيد هذه الخاصة معلومات المستخدم الحالي إذا تم التحقق من هوية المستخدم.

توجيه المستخدم إلى صفحة جديدة:

إذا عدنا للمثال الذي عملنا عليه في الجزء الأول من هذه الجلسة و الخاص بتطبيق محول العملات سنلاحظ أن العمل كله قد تم ضمن صفحة واحدة.

سنتعلم في هذا الجزء كيفية الملاحة من صفحة إلى أخرى و إعادة توجيه المستخدم.

هناك عدة طرق لنقل المستخدم من صفحة إلى أخرى.

تعد الطريقة الأسهل لتحقيق هذا الغرض هي استخدام تأشيرة <a> التي تحول النص الذي تحصره إلى وصلة تشعبية في هذا المثال here هي الوصلة إلى الصفحة الأخرى:

```
Click <a href="newpage.aspx">here</a> to go to newpage.aspx.
```

يتوفر خيار آخر لإرسال المستخدم إلى صفحة أخرى هو استخدام الرماز.

تعد هذه المقاربة أفضل عندما نريد من النص البرمجي أن يقوم بعملية ما قبل إعادة توجيه المستخدم أو في حال أردنا الرماز أن يقرر أين سيتم توجيه المستخدم (حالة كون النص البرمجي يتأكد من كون المستخدم جديد فيحوله إلى صفحة التسجيل أو مستخدم حالي فيحوله إلى صفحة الشراء مثلاً.)

يمكنك الاستفادة من مقبض الحدث ServerClick للتحكم بتحويل المستخدم إلى صفحة أخرى.

يمكنك الوصول إلى غرض HttpResponse الحالي عبر الخاصة Page.Response.

المثال التالي يقوم بإرسال المستخدم إلى صفحة أخرى على نفس مجلد الموقع:

```
Response.Redirect("newPage.aspx");
```

عند استخدام الطريقة Redirect توقف ASP.Net فوراً عملية معالجة الصفحة و يتم إرسال إعادة توجيه إلى الزبون.

لن يتم تنفيذ أي رماز برمجي ضمن الصفحة بعد الطريقة Redirect.

```
Response.Redirect("http://www.prosetech.com");
```

كما و يمكن استخدام الطريقة Transfer الخاصة بكائن HttpServerUtility.

تتميز هذه الطريقة بأنها لا ترسل رسالة إعادة توجيه إلى المستعرض و لكن تقوم بنقل التنفيذ إلى صفحة أخرى على المخدم وكأن المستخدم طلب هذه الصفحة أصلاً.

إن محدودية هذه الطريقة تتمثل في عدم قدرتها على تحويل المستخدم إلى صفحة خارج الموقع أو صفحة ليست صفحة ASP.NET كصفحة Html مثلاً.

ترميز HTML:

كما نعلم هناك مجموعة من المحارف المعينة التي لها معاني خاصة في HTML فعلى سبيل المثال تستخدم محارف إشارتي أكبر وأصغر (<>)لإنشاء التأشيرات.

قد تسبب هذه المحارف مشكلة في حال أردنا استخدامها كجزء من المحتوى على صفحة الوب

مثلاً إذا أردنا كتابة العبارة

Enter a word <here>

ستكون النتيجة في حال تم استخدام هذا النص لوضعه على الصفحة

Enter a word

لتجنب تفسير هذه المحارف الخاصة و تقوم بإظهارها تماماً كما تمت كتابتها يمكنك استخدام الخاصة InnerText التي قمنا بشرحها.

و لكن ماذا لو لم توفر هذه التأشيرة الخاصة InnerText؟

الحل في هذه الحالة اعتماد الطريقة ()HtmlEncode للغرض HttpServerUtility

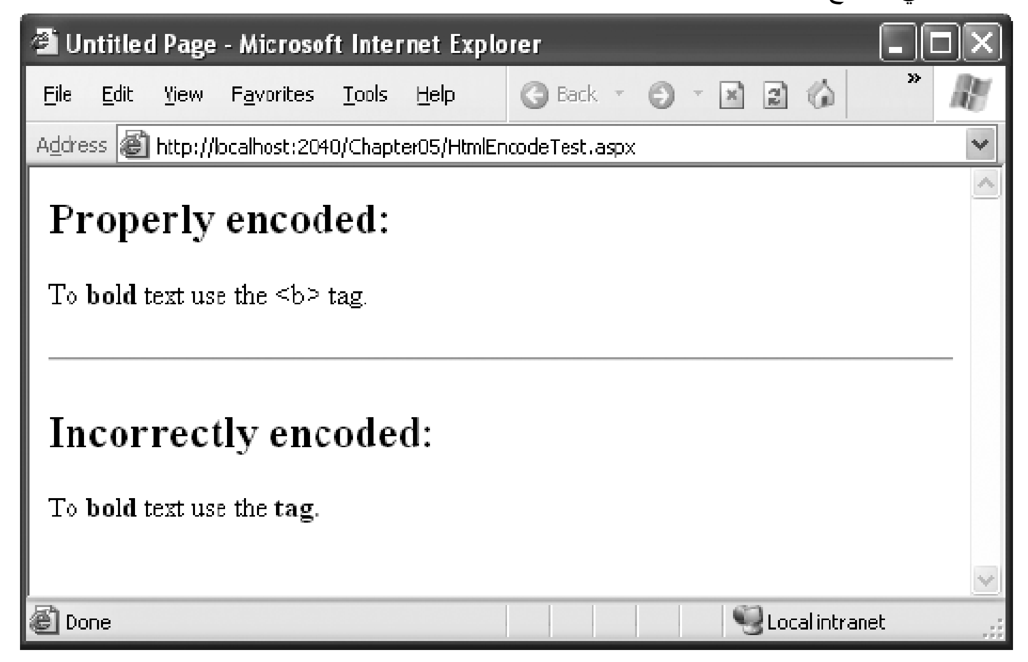
مثال (استخدام InnerHtml):

```
// Will output as "Enter a word &lt;here&gt;" in the HTML file, but the
// browser will display it as "Enter a word <here>".
ctrl.InnerHtml = Server.HtmlEncode("Enter a word <here>");
```

مثال (استخدام HtmlEncode):

```
ctrl.InnerHtml = "To <b>bold</b> text use the ";
ctrl.InnerHtml += Server.HtmlEncode("<b>") + " tag.";
```

الشكل التالي يوضح نتيجة تنفيذ المثالين السابقين:



هذه المقاربة مفيدة جداً عندما نريد عرض بيانات تمت استعادتها من قاعدة البيانات.

يمكننا إن أردنا إعادة النص من جديد إلى وضعه الأساسي قبل تحويل المحارف الخاصة باستخدام الطريقة ()HtmlDecode.

يوفر الغرض HttpServerUtility طريقتين مشابهتين مفيدتين أيضاً هما ()UrlEncode و ()UrlDecode يمكن استخدام هاتين الطريقتين عند تمرير معلومات بين الصفحات ضمن ترويسة طلب Http بإضافتهاإلى عنوان URL.

أحداث التطبيق:

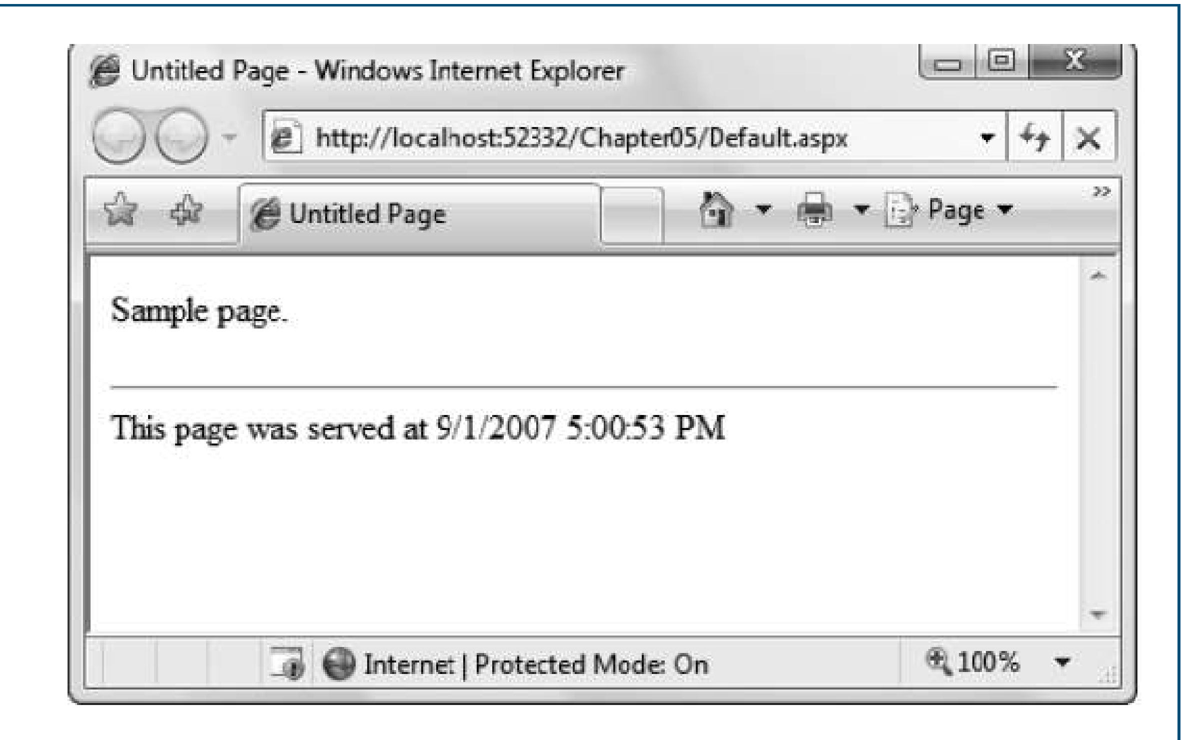
خلال سياق هذه الجلسة رأينا كيف توفر ASP.NET إمكانية معالجة الأحداث من خلال كتابة رماز مقبض الحدث.
و مع أن عناصر التحكم هي المصدر الأساسي للأحداث لكن هناك نوع آخر من الأحداث الذي نتعامل معه أحياناً .
هذه الأحداث هي أحداث التطبيق.
تستخدم أحداث التطبيق لأداء أعمال تتعلق بجميع الصفحات في التطبيق مثلاً يمكن أن يكتب رماز تسجيل الدخول ضمن مقبض حدث التطبيق ليتم استدعاؤه في كل مرة يتم فيها استقبال طلب على صفحة مهما كانت تلك الصفحة.
لا يمكنك معالجة أحداث التطبيق في الرماز في الخلفية و تحتاج لإتمام هذه العملية إلى مكون جديد هو ملف Global.asax.

الملف Global.asax:

يسمح الملف Global.asax لنا بكتابة النص البرمجي للاستجابة لأحداث التطبيق. يتم إطلاق هذه الأحداث أثناء مراحل العمل المختلفة.
لإضافة ملف Global.asax إلى التطبيق في Visual Studio اختر WebSite-> add New Item
ثم قم باختيار global.asax ثم اضغط على خيار OK.
إن ملف Global.asax مشابه من حيث الشكل لملف aspx بفرق كون الأخير لا يستطيع احتواء تأشيرات HTML أو ASP.NET و يحتوي بدلاً عنها مقابض أحداث ، على سبيل المثال يحدد ملف Global.asax التالي استجابة لحدث Application.EndRequest الذي يتم إطلاقه في كل مرة قبل إرسال صفحة إلى المستخدم:

```
<%@ Application Language="C#" %>
<script language="c#" runat="server">
protected void Application_OnEndRequest()
{
Response.Write("<hr />This page was served at " +
DateTime.Now.ToString());
}
</script>
```

يقوم هذا المقبض باستخدام الطريقة Write() لغرض Response لإضافة تذييل إلى أسفل الصفحة تبين التاريخ و الوقت الذي تم فيه خلق الصفحة.



يمكن لكل تطبيق ASP.NET أن يحتوي ملف Global.asax وحيد يتم التعرف آلياً عليه و استخدامه في حال وجوده ضمن مجلد التطبيق.

المزيد من أحداث التطبيق:

هناك الكثير من الأحداث الخاصة بالتطبيق بالإضافة إلى الحدث Application.EndRequest.
لإنشاء مقابض الأحداث يكفي أن تعريف الإجرائيات الفرعية بالاسم المحدد.
فيما يلي جدول يتضمن أسماء أهم طرق مقابض الأحداث ووصف لعملها:

Application_Start()	يتم إطلاق هذا الحدث في المرة الأولى التي يتم فيها استقبال طلب من أي مستخدم و لمرة واحدة. يستخدم هذا المقبض عادة لتأهيل بعض المعلومات الأولية التي سيتم إعادة استخدامها لاحقاً.
Application_End()	يتم إطلاق هذا الحدث عند إنهاء عمل التطبيق عادة بسب إعادة تشغيل مخدم الوب ، يمكن في هذا المقبض استخدام رماز بمهمة إجراء عمليات تنظيف و تحرير للموارد المحجوزة.
Application_BeginRequest()	يتم إطلاق هذا الحدث مع كل طلب يستقبله التطبيق قبل تنفيذ أي صفحة.
Application_EndRequest()	يتم إطلاق هذا الحدث مع كل طلب يستقبله التطبيق تماماً بعد تنفيذ رماز الصفحة.
Session_Start()	يتم إطلاق هذا الحدث عندما يتم استقبال طلب من مستخدم جديد (نقصد بمستخدم جلسة جديدة).

Session_End()	يتم إطلاق هذا الحدث عند انتهاء زمن الجلسة أو عند إنهائها برمجياً ، يتم إطلاق هذا الحدث فقط عندما تستخدم خيار InProc لتخزين الجلسة و ليس StateServer أو SQLServer.
Application_Error()	يتم إطلاق هذا الحدث تجاوباً مع الأخطاء التي لا تتم معالجتها.

إعدادات ASP.NET:

الموضوع الأخير الذي سيتم تناوله في هذه الجلسة هو ملف الإعداد في ASP.NET.
يحتوي كل تطبيق وب على ملف web.config الهدف من هذا الملف هو تشكيل الإعدادات الأساسية للتطبيق ابتداءاً من طريقة التعامل مع رسائل الخطأ و حتى إعدادات الأمن الخاصة بحجب الزوار الغير مرغوب بهم.
لاستخدام الملف web.config هناك العديد من المميزات أهمها:

1. غير مغلق: أي يمكن تحديث ملف الإعدادات حتى أثناء عمل التطبيق حيث ستحتفظ الطلبات التي تم استقبالها قبل التعديل بنفس الإعداد في حين تخدم الطلبات القادمة بعد التعديل وفق الإعداد الجديد.
2. يمكن الوصول إليها بسهولة و نسخها: مثلاً يمكن الوصول إليها عبر حاسب بعيد في حال توفرت الصلاحيات المناسبة ، كما يمكن نسخها بغرض تطبيقها على تطبيق آخر.
3. يمكن فهم محتواها و تحريرها بسهولة : تعتبر الإعدادات المستخدمة قابلة للقراءة المباشرة من المطور بسهولة كما يمكن تعديلها بواسطة أي محرر نصوص عادي دون الحاجة إلى أدوات تشكيل خاصة

ملف Web.config:

يستخدم ملف web.config تنسيق XML بتأشيرات مسبقة التعريف. كامل محتوى الملف متضمن داخل تأشيرة الجذر <configuration>. يوجد داخل هذه العنصر مجموعة من الأقسام الفرعية ، بعضها يندر تغييره و البعض الآخر هام جداً ويستخدم بكثرة.
فيما يلي الهيكيلية الأساسية لملف web.config موضحاً فيه الأقسام الفرعية الأهم:

```
<?xml version="1.0" ?>
<configuration>
<configSections>...</configSections>
<appSettings>...</appSettings>
<connectionStrings>...</connectionStrings>
<system.web>...</system.web>
<system.codedom>...</system.codedom>
<system.webServer>...</system.webServer>
</configuration>
```

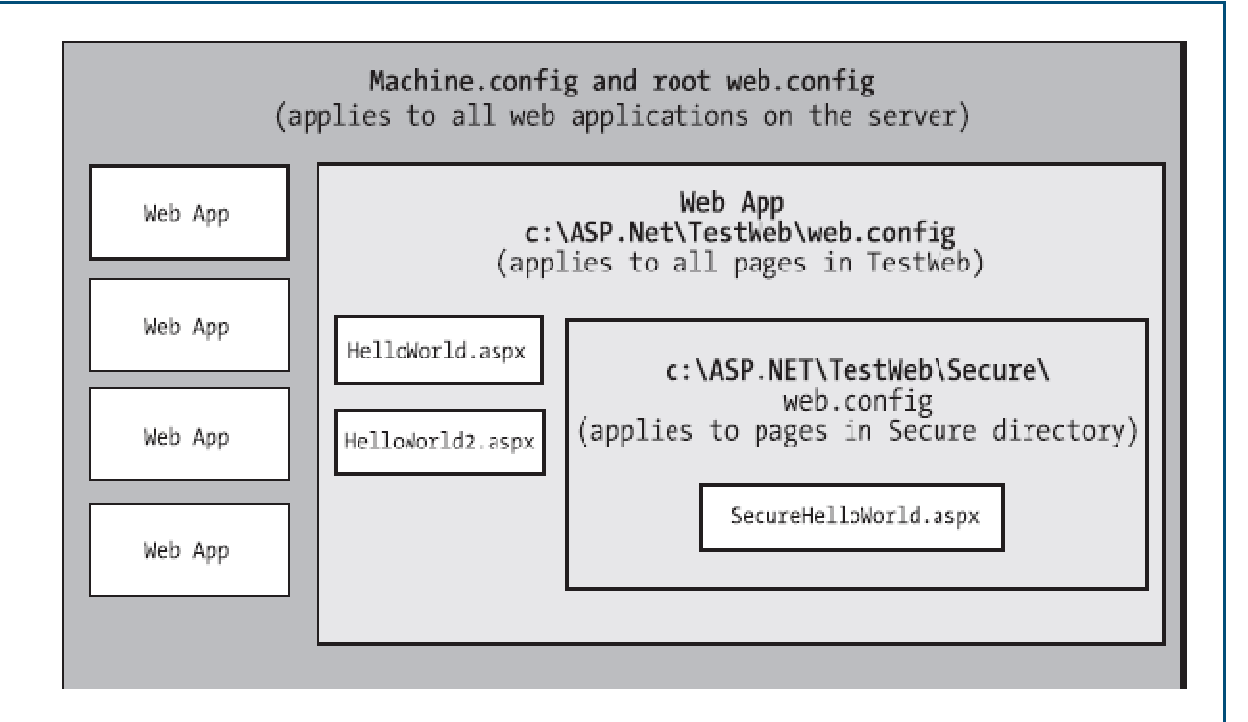
Note that the web.config file is case-sensitive, like all XML documents, and starts every setting with a lowercase letter. This means you cannot write <AppSettings> instead of <appSettings>.

يفيد الجزء <appSetting> في إضافة معلومات عامة يمكنك استخدامها ضمن التطبيق. في حين تم تخصيص القسم الفرعي <connectionString> لتعريف ممعلومات الاتصال إلى قاعدة المعطيات.
أما القسم الفرعي <system.web> فهو مسؤول عن ضبط إعدادات ASP.NET حيث يتوفر ضمن هذ القسم عناصر تتعلق بكل ناحية من نواحي إعدادات التطبيق.
إذا أردنا مثلاً تخصيص إعدادات الأخطاء يكفي إضافة التأشيرة <customErrors> ضمن تأشيرة<system.web>
إذا أردنا التحكم بإعدادات الأمان في ASP.NET يمكن التفكير بإضافة التأشيرة <authentication> ضمن القسم <authorization>.

الإعداد بطريقة الاحتواء:

تستخدم ASP.NET إعدادات تعتمد تعددية الطبقات التي تسمح بتعيين الإعدادات على أكثر من سوية.
يبدأ أي مخدم وب بإعدادات رئيسية يتم تعريفها ضمن ملفين في المجلد
c:\Windows\Microsoft.NET\Framework\v2.0.50727\Config هذان الملفان هما machine.config و web.config.
لن تقوم بتحرير محتوى أي من هذين الملفين بشكل عام بشكل يدوي لأنهما يؤثران على الجهاز كاملاً ، عوضاً عن هذا سنقوم بتحرير الإعدادات لملف web.config الموجود في مجلد التطبيق الخاص بنا. حيث يمكننا إضافة معلومات جديدة أو إعادة تعريف الإعدادات الافتراضية التي تم تحديدها ضمن الملفين السابقين.
و المثير للاهتمام أكثر أنه بإمكاننا حتى تعديل الإعدادات لأجزاء مختلفة من التطبيق .لهذا الغرض لا بد من إنشاء مجلدات فرعية تستطيع احتواء ملف web.config الخاص بها.

لنفرض مثلاً أننا قمنا بإنشاء مجلد ضمن الفهرس c:\ASP.NET\TestWeb\Secure فإن بإمكانه الاستحصال على المعلومات من ثلاث ملفات كما يظهر في الشكل التالي:



كما نلاحظ أن الإعدادات الموجودة ضمن أي ملف machine.config و web.config سيتم إعتمادها ما لم يتم إعادة تعريفها. بهذه الطريقة لسنا بحاجة لإعادة تعيين الإعدادات كاملة في كل سوية و يكفي إعادة تعريف الإعدادات الخاصة بالتعديلات المراد تطبيقها على محتوى المجلد المحدد.

تخزين الإعدادات المخصصة ضمن ملف web.config:

تمكن ASP.NET من عملية تخزين أية إعدادات خاصة إضافية يرغب بها المطور ضمن ملف web.config في العنصر المسمى <appSetting> حيث يكون هذا العنصر محتوى ضمن العنصر الجذر لملف web.config وهو <configuration>.
النص التالي يبين هيكلية هذا الملف:

```
<?xml version="1.0" ?>
<configuration>
...
<appSettings>
<!-- Custom application settings go here. -->
</appSettings>
...
<system.web>
<!-- ASP.NET Configuration sections go here. -->
</system.web>
...
</configuration>
```

تكون الإعدادات الخاصة التي تود إضافتها مكتوبة كمتحول سلسة محرفية بسيط.

يمكن أن نحتاج إلى إعدادات خاصة في web.config لعدة أسباب:

1. مركزية الإعدادات الهامة التي تستخدم في أكثر من صفحة. مثلاً قد تقوم بإنشاء متحول يقوم بتخزين متحول يحتوي استعلام من قاعدة البيانات عندها تستطيع أي صفحة الوصول إلى هذه القيمة باستعادتها من ملف web.config
2. لتمكين طريقة سهلة و سريعة للتحويل بين أنماط تشغيل مختلفة للتطبيق.
3. لإعطاء قيم أولية لبعض العناصر ، حيث يقوم المستخدم بتغييرها إن رغب أثناء التشغيل و لكنها توفر القيمة الافتراضية.

مثلاً يمكننا إدراج إعدادات مخصصة باستخدام العنصر <add> الذي يقوم بتعريف اسم متحول و محتوى كما في النص التالي:

```
<appSettings>
<add key="DataFilePath"
value="e:\NetworkShare\Documents\WebApp\Shared" />
</appSettings>
```

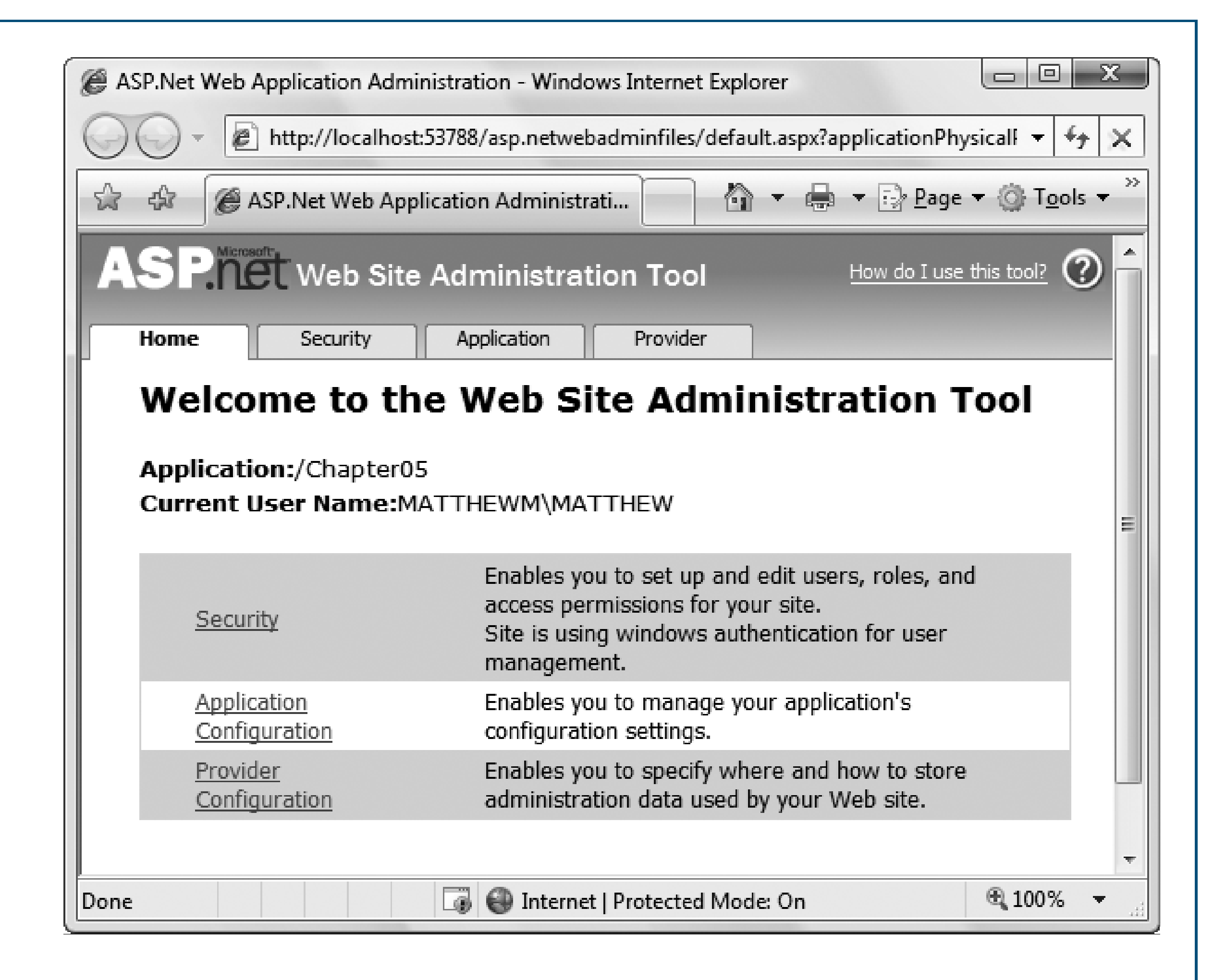
للوصول إلى الإعدادات المخصصة التي قمنا بتحديدها عن طريق الصف WebConfiguration الموجود ضمن الصف:

```
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.Configuration;
public partial class ShowSettings : System.Web.UI.Page
{
protected void Page_Load()
{
lblTest.Text = "This app will look for data in the directory:<br /><b>";
lblTest.Text += WebConfigurationManager.AppSettings["DataFilePath"];
lblTest.Text += "</b>";
}
}
```

ملاحظة: تم إعداد ASP.NET لترفض بشكل تلقائي أي طلب للوصول إلى ملفات config. بمعنى أن أي مستخدم بعيد لن يتمكن من الوصول إلى الملف عن طريق مخدم IIS.

أداة إدارة الموقع (WAT)

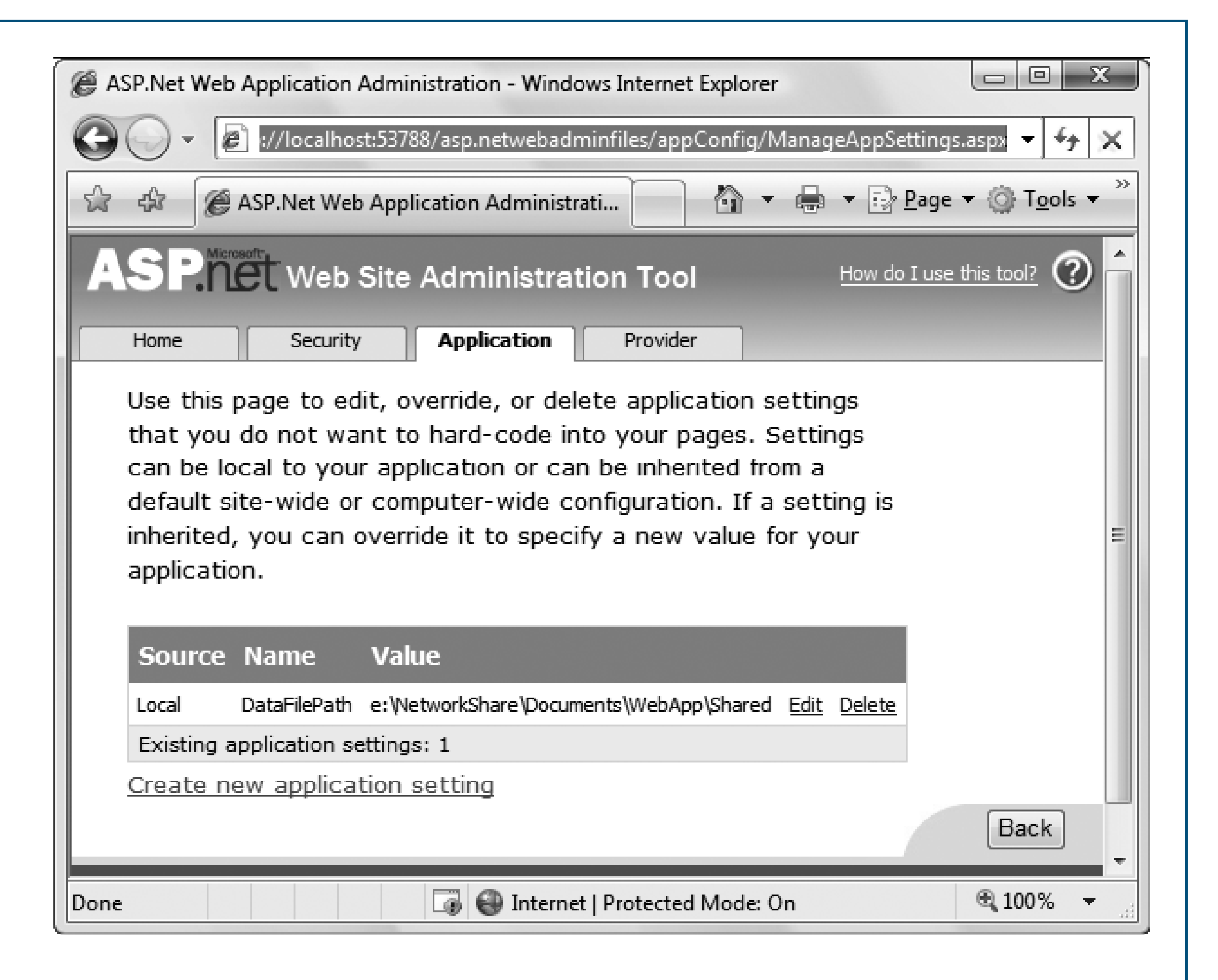
إن تحرير ملف web.config بشكل مباشر يمكن أن يصبح عملية مضجرة.لذلك و لجعل عملية التحرير أكثر مرونة قامت ASP.NET بتضمين أداة إعداد خاصة بواجهة رسومية تسمح بالعمل على الأقسام المختلفة لملف web.config.



يمكن استخدام هذه الأداة لأتمتة التغيرات على ملف web.config.

توفر الأداة إمكانية ضبط الإعدادات العامة للتطبيق إضافة إلى تلك المتعلقة بالأمان و الاتصال بقواعد البيانات.

فإذا أردنا تعيين قيمة للمتحول DataFilePath كما فعلنا يدوياً في المثال السابق يكفي الوصول إلى الأداة عن طريق خيار، ASP configuration. ثم اختيار التبويب Application و النقر على الوصلة Create new application setting وإدخال اسم المتحول و القيمة الخاصة به.



ستقوم هذه الأداة تلقائياً بتوليد وإضافة إعداداتنا الجديدة على ملف web.config الخاص بالتطبيق.

الفصل السابع والثامن

عناصر تحكم الوب

يحصل الطالب في نهاية هذه الجلسة على فهم لـــ:

1. ماهية عناصر الوب
2. الاختلاف بين عناصر الوب و عناصر تحكم HTML
3. تأشيرات عناصر الوب الأساسية ،الخصائص و الأحداث
4. الصفوف القاعدية الرئيسية لعناصر الوب و بنيتها
5. الترقيم و استخدامه للتعامل مع قيم الخصائص لعناصر الوب المختلفة

عناصر تحكم الوب:

تم في الجلسة الماضية التقديم للنموذج الذي تعتمده ASP.NET في البرمجة الموجهة بالأحداث.

يعتمد هذا النموذج على عناصر التحكم ويسمح ،كما شاهدنا، بتطوير تطبيقات الوب بالاعتماد على نفس التقنيات التي تستخدم عادة لبناء تطبيقات windows العادية.

تقدم عناصر تحكم HTML من جهة المخدم جزءاً بسيطاً جداً من الفوائد التي يقدمها نموذج البرمجة من جهة المخدم في ASP.NET. ولكن للوصول إلى الميزات الحقيقية لـــ ASP.NET لا بد لنا من استخدام عناصر تحكم أكثر مرونة وقابلية للتطوير. هذه العناصر هي عناصر تحكم الوب.

لفهم عناصر التحكم و استعمالها بشكل تطبيقي سنعتمد مثال (تطبيق بناء بطاقة تهنئة الكترونية على صفحة الوب) وسنحاول تطبيق معظم المعارف المكتسبة على هذا المثال.

لماذا يجب أن ننتقل إلى استخدام عناصر تحكم وب عوضاً عن عناصر تحكم HTML من جهة المخدم:

- تم بناء عناصر تحكم الوب بحيث توفر واجهة غنية للمستخدم دون أن تكون بالضرورة مقابلة لأحد العناصر المستخدمة في لغة HTML.

  مثلاً ،إذا قمنا باستخدام عنصر تحكم Calendar وGridVIew سنحصل على خرج صفحة يتضمن عشرات من تأشيرات الـــHTML.
- في عناصر تحكم الوب لسنا بحاجة لأن نعرف أي شيء عن HTML لأن عنصر التحكم ينشئ تأشيرات Html نيابة عنّا.
- توفر عناصر تحكم الوب نموذج أغراض متجانس: بعكس ما رأيناه في عناصر تحكم HTML من جهة المخدم .فإذا أخذنا على سبيل المثال علبة الإدخال البسيطة سنلاحظ بأننا سنحتاج إلى ثلاثة عناصر تحكم HTML للحصول على الوظيفة المناسبة وهي <input type="text"> و<input text="Password"> وعنصر التحكم <textarea>، في حين يتوفر عنصر تحكم وب واحد يوفر كل هذه الإمكانيات.
- تقوم عناصر تحكم الوب بضبط الخرج بشكل آلي،حيث يمكن لعناصر تحكم الوب أن تتعرف على أنواع المستعرضات وتقوم بعملية ضبط آلي لخرج Html المتولد لتتمكن من الاستفادة من المزايا التي يوفرها المستعرض (مثل دعم Javascript).

- توفر عناصر تحكم الوب مزايا عالية المستوى قياساً مع تلك المتوفرة في عناصر تحكم HTML. حيث توفر هذه العناصر المزيد من الأحداث و الخصائص و الطرق.

صفوف عناصر تحكم وب الرئيسية:

يحتوي الجدول التالي قائمة بصفوف عناصر تحكم وب الأساسية وتأشيرات HTML التي تولدها.
بعض عناصر التحكم هذه ،كما ذكرنا، يتم تصبيرها أحياناً بأكثر من تأشيرة. كما أن البعض منها أيضاً لا يملك مرادفات في HTML مثل عناصر تحكم CheckBoxList أو RadioButtonList فالخرج الخاص بها هو تركيبة من أكثر من تأشيرة. يتم تصبير عناصر التحكم السابقة عادة كتأشيرة جدول <table> تحتوي مجموعة من تأشيرات <input type="checkbox"> أو <input type="radio">.

صف عنصر التحكم	تأشيرة HTML الموافقة
Label	<span>
Button	<input type="submit"> or <input type="button">
TextBox	<input type="text">, <input type="password">, or <textarea>
CheckBox	<input type="checkbox">
RadioButton	<input type="radio">
Hyperlink	<a>
LinkButton	<a> with a contained <img> tag
ImageButton	<input type="image">
Image	<img>
ListBox	<select size="X">حيث يمثل X عدد الصفوف التي يتم إظهارها من خيارات اللائحة.
DropDownList	<select>
CheckBoxList	قائمة مكونة من تأشيرة <table> إضافة إلى مجموعة من تأشيرات <input type="checkbox">
RadioButtonList	قائمة مكونة من تأشيرة <table> إضافة إلى مجموعة من تأشيرات <input type="radio">
BulletedList	An ordered list (numbered) or unordered list (bulleted)

Panel	<div>
Table, TableRow, and TableCell	<table>, <tr>, and <td> or <th>

تأشيرات عناصر تحكم الوب:

لتأشيرات عناصر تحكم الوب تنسيق خاص فهي تبدأ دائماً بالسابقة :asp متبوعة باسم الصف.
في حال لم تكن التأشيرة تستلزم وسم إغلاق يجب أن تنتهي بـــ /> (تم اشتقاق هذه الصيغة من قواعد كتابة XML).
كل واصفة ضمن التأشيرة تعبّر عن خاصة لعنصر التحكم إلا الواصفة "runat="server التي تحدد كون عملية معالجة عنصر التحكم ستتم على المخدم.

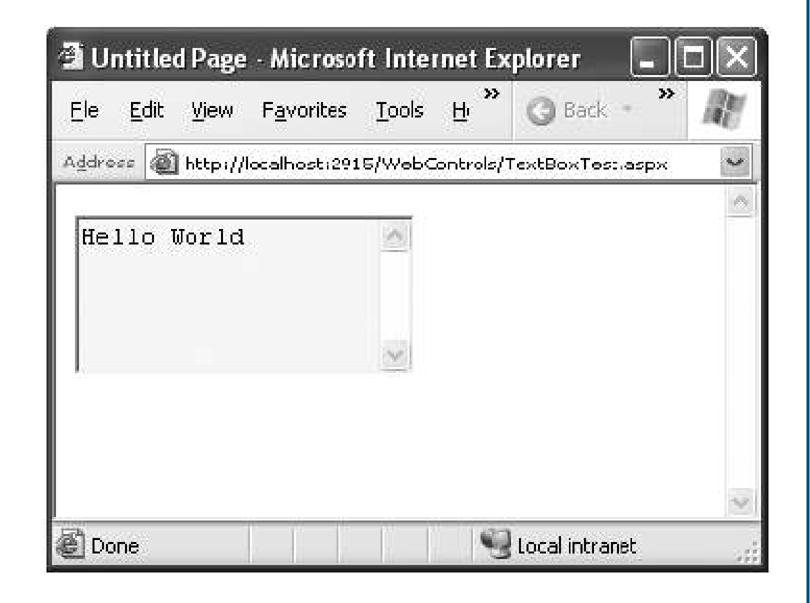
```
<asp:TextBox ID="txt" runat="server" />
```

عندما يقوم الزبون بطلب الصفحة الحاوية على عنصر التحكم هذا سيتم إعادة خرج HTML التالي:

```
<input type="text" ID="txt" name="txt" />
```

يمكنك التحكم بخصائص عنصر التحكم هذا كتعيين قيمة للنص وتحديد حجمه أو جعله مخصص للقراءة فقط وتغيير لون الخلفية.
تسمح الخاصة TextBox.TextMode مثلاً بتحديد كون عنصر التحكم سيعمل كعنصر إدخال نص بسطر وحيد أو متعدد الأسطر أو كعلبة نص مخصصة لإدخال كلمة السر.
يمكننا تحديد اللون باستخدام خصائص BackColor و ForColor و تعديل حجم علبة النص باستخدام الخاصة Rows.
يظهر النص البرمجي التالي مثالاً عن علبة نص مخصصة:

```
<asp:TextBox ID="txt" BackColor="Yellow" Text="Hello World"
ReadOnly="True" TextMode="MultiLine" Rows="5" runat="server" />
```



أما نص HTML الناتج عند تصيير عنصر التحكم هذا فهو على الشكل:

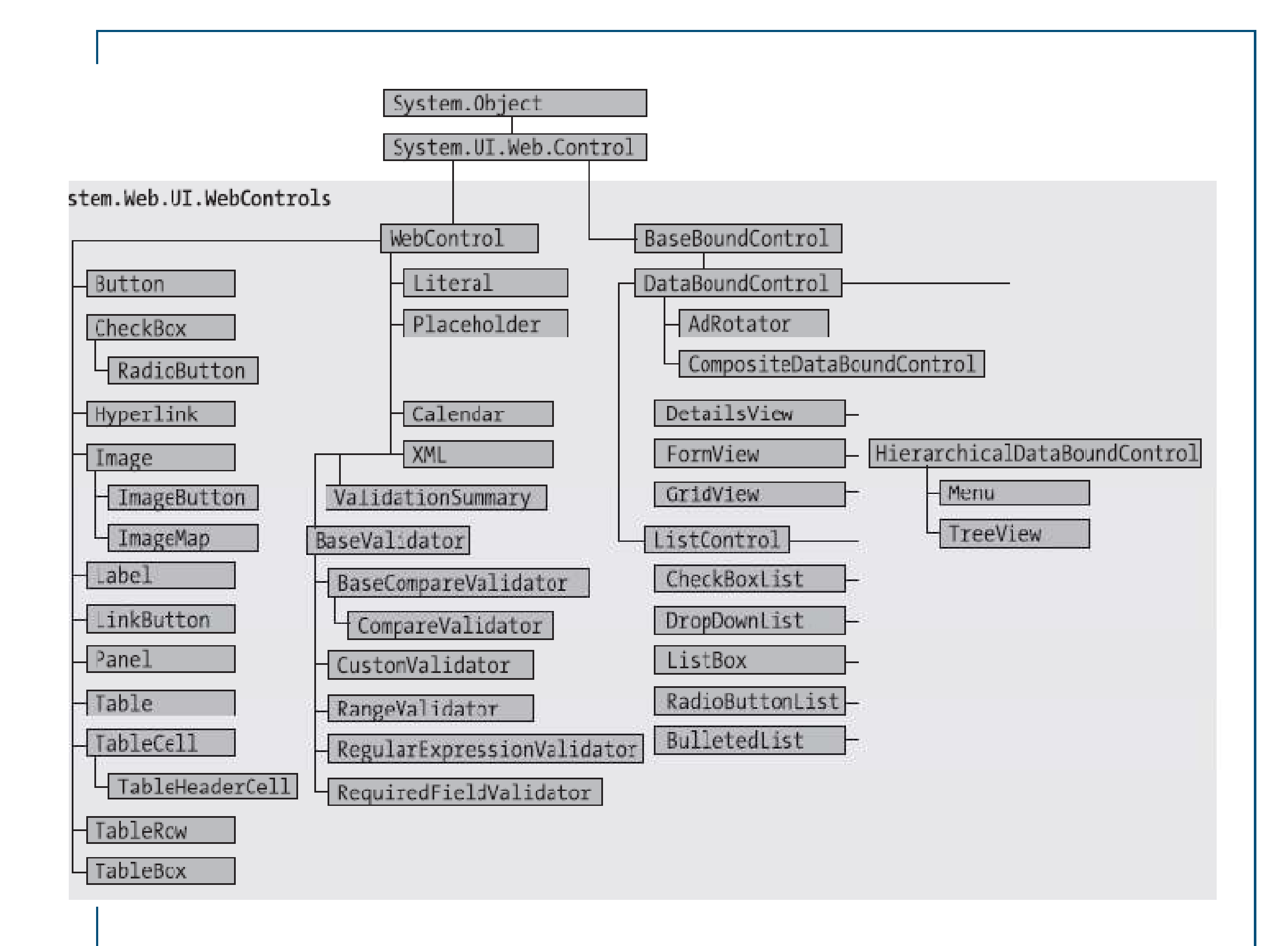
```
<textarea name="txt" rows="5" cols="20" readonly="readonly" ID="txt"
style="background-color:Yellow;">Hello World</textarea>
```

ملاحظة هامة: ASP.NET غير حساسة لحالة الأحرف في الجزء الخاص بتأشيرات عناصر التحكم.

صفوف عناصر تحكم الوب:

تعرّف عناصر تحكم الوب ضمن فضاء الأسماء System.Web.UI.WebControls وتتبع هرمية مختلفة عن تلك الخاصة بعناصر تحكم HTML .

يظهر الشكل التالي هرمية عناصر تحكم الوب:



صف WebControl القاعدي:

ترث معظم عناصر تحكم الوب الصف القاعدي WebControl ،حيث يحدد هذا الصف الوظائف الأساسية مثل المهام الخاصة بعملية الربط مع قواعد البيانات.

يوضح الجدول التالي أهم خصائص هذا الصف:

الخاصة	الوصف
AccessKey	تحدد هذه الخاصة الاختصار لنقل التركيز إلى عنصر التحكم. تأخذ هذه الخاصة قيمة مكونة من محرف واحد، حيث يؤدي ضغط هذا المحرف إضافة إلى زر ALT إلى انتقال التركيز إلى العنصر.
BackColor, ForeColor, and BorderColor	تحدد هذه الخواص لون الخلفية ،الكتابةو لون الإطار
BorderWidth	تحدد هذه الخاصة سماكة الإطار الخاص بعنصر التحكم.
BorderStyle	يمكن لهذه الخاصة أن تأخذ إحدى القيم مسبقة التعريف و التي

	تحدد نمط الإطار ,Dashed,Dotted,Double Groove,Ridge,Inset,Outset,Solid و None
Controls	تعيد هذه الخاصة مرجع إلى مجموعة جميع عناصر التحكم التي يحتويها عنصر التحكم هذا. إن مرجع الغرض الذي تتم إعادته ينتمي إلى الصف القاعدي System.Web.UI.Control. لذا لا بد من تحويله في حال الرغبة بالوصول إلى الخصائص المميزة لعنصر التحكم المحتوى.
Enabled	عند إسناد القيمة false إلى هذه الخاصة سيبقى عنصر التحكم هذا مرئياً ولكن لن يستطيع تقبّل أي دخل من أو الحصول على التركيز.
EnableViewState	عند إسناد القيمة false إلى هذه الخاصة يتم إيقاف تفعيل إدارة الحالة التلقائية لعنصر التحكم . ستعود في هذه الحالة جميع خصائص عنصر التحكم إلى قيمتها البدائية في كل مرة يتم فيها إرسال الصفحة.. التي تم تحديدها في الواصفة الخاصة بتأشيرة عنصر التحكم. أما إذا تم إسناد القيمة True إلى هذه الخاصة فسيقوم عنصر التحكم باستخدام حقل إدخال مخفي. لتخزين معلومات عن الخصائص. تضمن هذه العملية إمكانية الاختفاظ بأية تعديلات تم إجراؤها عن طريق الرماز على خصائص عنصر التحكم
Font	تحدد هذه الخاصة نوع الخط المستخدم في تصيير أي نص ضمن عنصر التحكم .
Height and Width	تحدد هذه الخاصة ارتفاع و عرض عنصر التحكم. يتم إهمال قيم هذه الخصائص في بعض عناصر التحكم عند استخدامها مع نسخ قديمة من مستعرضات الوب.
Page	تعيد هذه الخاصة مرجع لصفحة الوب الذي يحتوي عنصر التحكم كغرض من الصف System.Web.UI.Page
Parent	تعيد هذه الخاصة مرجع إلى عنصر التحكم الأب االذي يحتوي عنصر التحكم . ستعيد هذه الخاصة مرجع إلى صفحة الوب الحالية في حال تم وضع عنصر التحكم بشكل مباشر ضمن الصفحة.
TabIndex	قيمة عددية تمكّن من عملية التحكم بتسلسل انتقال التركيز بين عناصر التحكم المختلفة لدى ضغط زر Tab
ToolTip	تحدد هذه الخاصة قيمة نصية يتم إظهارها عند وضع مؤشر الفأرة

	فوق العنصر.
Visible	عند إسناد القيمة false إلى هذه الخاصة سيختفي عنصر التحكم ولن يتم تصييره ضمن خرج HTML النهائي للصفحة التي سيتم إرسالها إلى الزبون.

الوحدات المستخدمة:

تستخدم البنية Unit في جميع وحدات القياس مثل BorderWidth, Height, Width.

تجمع هذه البنية القيمة العددية مع نوع القياس مثل (pixel,percentage ...)، لذلك يجب عند إدخال قيمة أحد هذه القياسات تحديد نوع القياس كما في المثال:

```
<asp:Panel Height="300px" Width="50%" ID="pnl" runat="server" />
```

أما عند استخدام هذه القيم ضمن الرماز فعلينا استخدام الطرق الساكنة التي يوفرها Unit كما يظهر في المثال التالي:

```
// Convert the number 300 to a Unit object
// representing pixels, and assign it.
pnl.Height = Unit.Pixel(300);
// Convert the number 50 to a Unit object
// representing percent, and assign it.
pnl.Width = Unit.Percentage(50);
```

كما يمكننا إنشاء غرض Unit ثم تأهيله بالبيانات كما في المثال:

```
// Create a Unit object.
Unit myUnit = new Unit(300, UnitType.Pixel);
// Assign the Unit object to several controls or properties.
pnl.Height = myUnit;
pnl.Width = myUnit;
```

الترقيم (Enumerations):

يستخدم الترقيم بشكل كبير في صفوف NET. لتجميع الثوابت ذات الصلة.

على سبيل المثال عند اسناد قيمة إلى الخاصة BorderStyle يمكننا الاختيار من مجموعة من القيم مسبقة التعريف للترقيم BorderStyle.

للوصول إلى ترقيم محدد من المجموعة نستخدم صيغة (النقطة .):

```
ctrl.BorderStyle = BorderStyle.Dashed;
```

أما ضمن ملف aspx فيتم تعيين قيمة الترقيم بسلسلة محرفية من أحد الخيارات المتاحة للترقيم.

```
<asp:Label BorderStyle="Dashed" Text="Border Test" ID="lbl" runat="server" />
```

الألوان:

تعيد الخاصة Color مرجع إلى غرض Color.ينتمي هذا الغرض إلى فضاء الأسماء System.Drwaing.

يمكننا تمثيل الألوان بعدة طرق:

- استخدام الطريقة ARGB(alpha,red,green,blue) حيث يمكننا تمثيل الألوان بإعطاء قيم للمركبات اللونية الثلاث التي تعبر عن شدة اللون الأحمر و الأخضر و الأزرق إضافة إلى الشفافية. تأخذ كل من هذه المركبات قيمة صحيحة بين 0 و 255.
- باستخدام أسماء الألوان مسبقة التعريف في NET.
- باستخدام أسماء الألوان المستخدمة في HTML،حيث يتم تعيين قيمة الألوان كسلسلة محرفية باستخدام الصف ColorTranslator.

المثال التالي يوضح الطرق المختلفة لتعيين خاصة اللون:

```
// Create a color from an ARGB value
int alpha = 255, red = 0, green = 255, blue = 0;
ctrl.ForeColor = Color.FromArgb(alpha, red, green, blue);
// Create a color using a .NET name
ctrl.ForeColor = Color.Crimson;
// Create a color from an HTML code
ctrl.ForeColor = ColorTranslator.FromHtml("Blue");
```

أما ضمن الملف aspx فيتم استخدام الصيغة:

```
<asp:TextBox ForeColor="Red" Text="Test" ID="txt" runat="server" />
```

أو صيغة مماثلة لتلك المستخدمة في HTML و التي تعبر عن اللون كرقم مكتوب بصيغة ستعشرية:

```
<asp:TextBox ForeColor="#ff50ff" Text="Test" ID="txt" runat="server" />
```

الخطوط:

توفر الخاصية Font مرجعاً إلى غرض FontInfo المعرف ضمن فضاء الأسماء System.Web.UI.WebControls حيث يملك كل غرض FontInfo مجموعة من الخصائص التي تحدد اسمه وحجم الخط والنمط المستخدم.

الخاصة	الوصف
Name	اسم الخط المستخدم
Names	مصفوفة تحتوي أسماء مجموعة من الخطوط. يقوم المستعرض بالتحقق من توفر الخطوط حسب الترتيب الذي ترد فيه ضمن هذه المصفوفة.و يقوم باستخدام أول خط مطابق للخطوط الموجودة على حاسب الزبون و يدعمها المستعرض.
Size	تعبر هذه الخاصة عن حجم الخط حيث يمكن أن يمثل بقيمة نسبية أو مطلقة.

Bold,Italic,Strikeout,Underline,overline	خصائص منطقية تطبق النمط المحدد عند إسناد القيمة True إليها.

يمكن إسناد قيم إلى هذه الخصائص ضمن الرماز بالصيغة:

```
ctrl.Font.Name = "Verdana";
ctrl.Font.Bold = true;
```

كذلك يمكن تحديد حجم الخط باستخدام النمط FontUnit:

```
// Specifies a relative size.
ctrl.Font.Size = FontUnit.Small;
// Specifies an absolute size of 14 pixels.
ctrl.Font.Size = FontUnit.Point(14);
```

أما في ملف aspx. فنستخدم الصيغة:

```
<asp:TextBox Font-Name="Tahoma" Font-Size="40" Text="Size Test" ID="txt" runat="server" />
```

```
<asp:TextBox Font-Name="Tahoma" Font-Size="Large" Text="Size Test" ID="txt"
runat="server" />
```

عند استخدام نوع خط غير متوفر لدى الزبون ،يتم استخدام الخط التلقائي للمستعرض.

لذا من المفضل استخدام أكثر من نوع خط مفصولة بفاصلة ،لكي يتم اعتماد تلك الخطوط، وحسب الترتيب، في حال عدم وجود الخط المطلوب على حاسب الزبون كما يظهر في المثال:

```
<asp:TextBox Font-Names="Verdana,Tahoma,Arial" Text="Size Test" ID="txt" runat="server" />
```

التركيز (focus):

بعكس عناصر تحكم HTML توفر عناصر تحكم الوب الخاصة Focus().

عند تصيير الصفحة يتم البدء عند عنصر التحكم الذي يملك التركيز.

بالإضافة إلى إمكانية استدعاء الطريقة Focus()،يمكن ضمن ملف aspx،استخدام الواصفة DefaultFocus الخاصة بتأشيرة النموذج:

```
<form DefaultFocus="TextBox2" runat="server">
```

لا يمكن للعناصر التي لا تقبل الدخل مثل Label أن تحصل على التركيز و لكن يمكن استخدام الخاصة AssociatedControlID لتحديد عنصر إدخال مرتبط وبهذه الطريقة يتم تمرير التركيز من عنصر تحكم Label إلى عنصر التحكم المرتبط كما يوضح المثال التالي:

```
<asp:Label AccessKey="2" AssociatedControlID="TextBox2" runat="server"
Text="TextBox2:" />
<asp:TextBox runat="server" ID="TextBox2" />
```

الزر التلقائي:

تمكَن ASP.NET من تحديد أحد الأزرار في الصفحة كزر تلقائي. يتم ضغط هذا الزر تلقائياً لدى ضغط زر الإدخال من لوحة المفاتيح.

يتم تحديد الزر التلقائي ضمن تأشيرة عنصر التحكم باستخدام الواصفة DefaultButton كما في المثال:

```
<form DefaultButton="cmdSubmit" runat="server">
```

يمكن أن يكون الزر التلقائي أياً من عناصر التحكم التي تتبنى الواجهة IButtonControl مثل عناصر تحكم Button , LinkButton, ImageButton.

عناصر تحكم القوائم:

تتضمن عناصر تحكم القوائم عناصر BulletedList , RadioButton , CheckBoxList , DropDownList , ListBox. تعمل جميع هذه العناصر مبدئياً بنفس الطريقة و لكن يتم تصييرها بشكل مختلف. فعلى سبيل المثال يتم تصيير عنصر التحكم ListBox كقائمة مستطيلة تظهر عدة بنود في حين يظهر عنصر تحكم DropDownList العنصر المختار فقط ، أما بالنسبة لعنصر تحكم CheckBoxList فهو مشابه لـــعنصر تحكم ListBox بفرق كون كل بند يتم تصييره كعنصر CheckBox .

و أخيراً عنصر تحكم BulletedList و هو العنصر الوحيد الذي لا يسمح باختيار عناصر لكن يتم تصيير البنود ضمنه كتسلسل من العناصر المرقمة أو المسبوقة بتعداد نقطي.

توفر جميع عناصر تحكم القائمة خاصة SelectedIndex التي تحدد البند المختار. مثلاً إذا تم اختيار البند الأول تكون قيمة هذا الدليل مساوية للصفر.

```
LisItem item;
Item=myListBox.SelectedItem;
```

عناصر تحكم القائمة متعددة الاختيار:

تسمح بعض عناصر تحكم القائمة باختيار أكثر من بند (لا تدعم DropDownList و RadioButtonList هذه الخاصة).

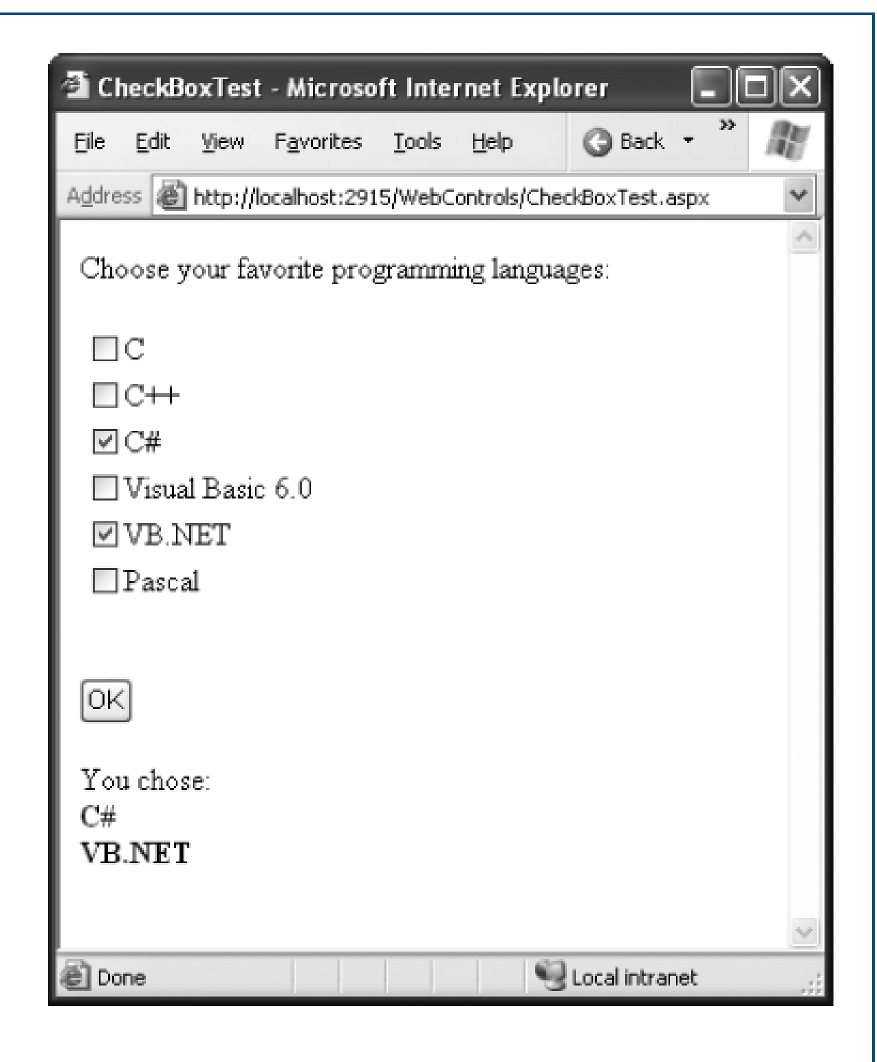
يتم تفعيل الاختار المتعدد باسناد القيمة المرقمة List.SelectionMode.Multiple إلى الخاصة SelectionMode .

يسمح عنصر التحكم عندها باختيار أكثر من بند من القائمة وذلك بضغط زر Ctrl ونقر البنود المراد اختيارها.

يتم الوصول إلى معلومات البنود المختارة من خلال المجموعة Items.

يمكن تحديد العناصر التي تم اختيارها عبر المرور من خلال حلقة على جميع عناصر هذه المجموعة و التحقق قيمة الخاصة ListItem.Selected لكل عنصر فيها.

يوضح المثال التالي استخدام عنصر تحكم CheckBoxList للحصول على خيارات اللغات التي قام المستخدم باختيارها.



يوضح النص التالي التأشيرات المستخدمة ضمن ملف aspx:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CheckListTest.aspx.cs" Inherits="CheckListTest" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>CheckBoxTest</title>
</head>
<body>
<form runat="server">
<div>
Choose your favorite programming languages:<br /><br />
<asp:CheckBoxList ID="chklst" runat="server" /><br /><br />
<asp:Button ID="cmdOK" Text="OK" OnClick="cmdOK_Click" runat="server" />
<br /><br />
<asp:Label ID="lblResult" runat="server" />
</div>
</form>
</body>
</html>
```

أما النص البرمجي الذي ستتم إضافته للتنقل بين بنود القائمة عند نقر زر OK فيكون على الشكل:

```
public partial class CheckListTest : System.Web.UI.Page
```

```
{
protected void Page_Load(object sender, EventArgs e)
{
if (!this.IsPostBack)
{
chklst.Items.Add("C");
chklst.Items.Add("C++");
chklst.Items.Add("C#");
chklst.Items.Add("Visual Basic 6.0");
chklst.Items.Add("VB.NET");
chklst.Items.Add("Pascal");
}
}
protected void cmdOK_Click(object sender, EventArgs e)
{
lblResult.Text = "You chose:<b>";
foreach (ListItem lstItem in chklst.Items)
{
if (lstItem.Selected == true)
{
// Add text to label.
lblResult.Text += "<br />" + lstItem.Text;
}
}
lblResult.Text += "</b>";
}
}
```

عنصر تحكم قائمة التعداد النقطي أو الرقمي:

يقابل عنصر التحكم هذا تأشيرة <ul> أو <ol> من جهة المخدم.

يملك عنصر التحكم هذا مجموعة من الخصائص تمكن من عملية التحكم بشكل إظهار هذا العنصر.

الجدول التالي يظهر بعض هذه الخصائص:

الخاصة	الوصف
BulletStyle	تحدد هذه الخاصة نمط القائمة سواء كانت رقمية أم قائمة تسلسل أبجدي أو قائمة أرقام رومانية أو بأشكال مثل الدائرة ، القرص ، المربع
BulletImageUrl	في حال تم إسناد قيمة CustomImage إلى الخاصة BulletStyle يمكن تحديد الصورة التي ستستخدم للتنقيط ضمن هذه الخاصة.
FirstBulletNumber	تستخدم هذه الخاصة مع التعداد الرقمي لتحديد العدد أو الحرف الذي سيبدأ الترقيم عنده.
DisplayMode	تحدد هذه الخاصة طريقة تصيير بنود القائمة (كنص أو كارتباط

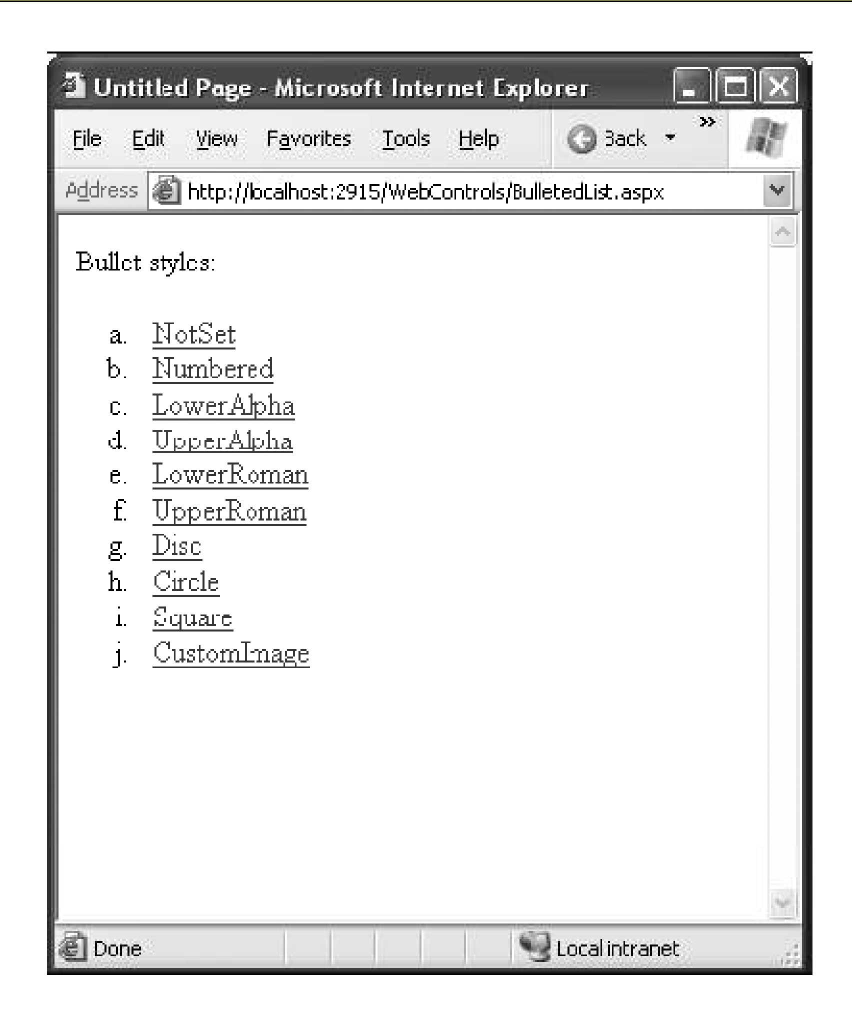
تشعبي).	

لتمكين الاستجابة للحدث Button.Click نقوم باسناد القيمة LinkButton إلى الخاصة Display mode.

يتم تحديد البنود التي تم اختيارها ضمن القائمة باستخدام غرض BulletedListEventArgs .

المثال التالي يوضح نص مقبض الحدث:

```
protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
{
string itemText = BulletedList1.Items[e.Index].Text;
Label1.Text = "You choose item" + itemText;
}
```

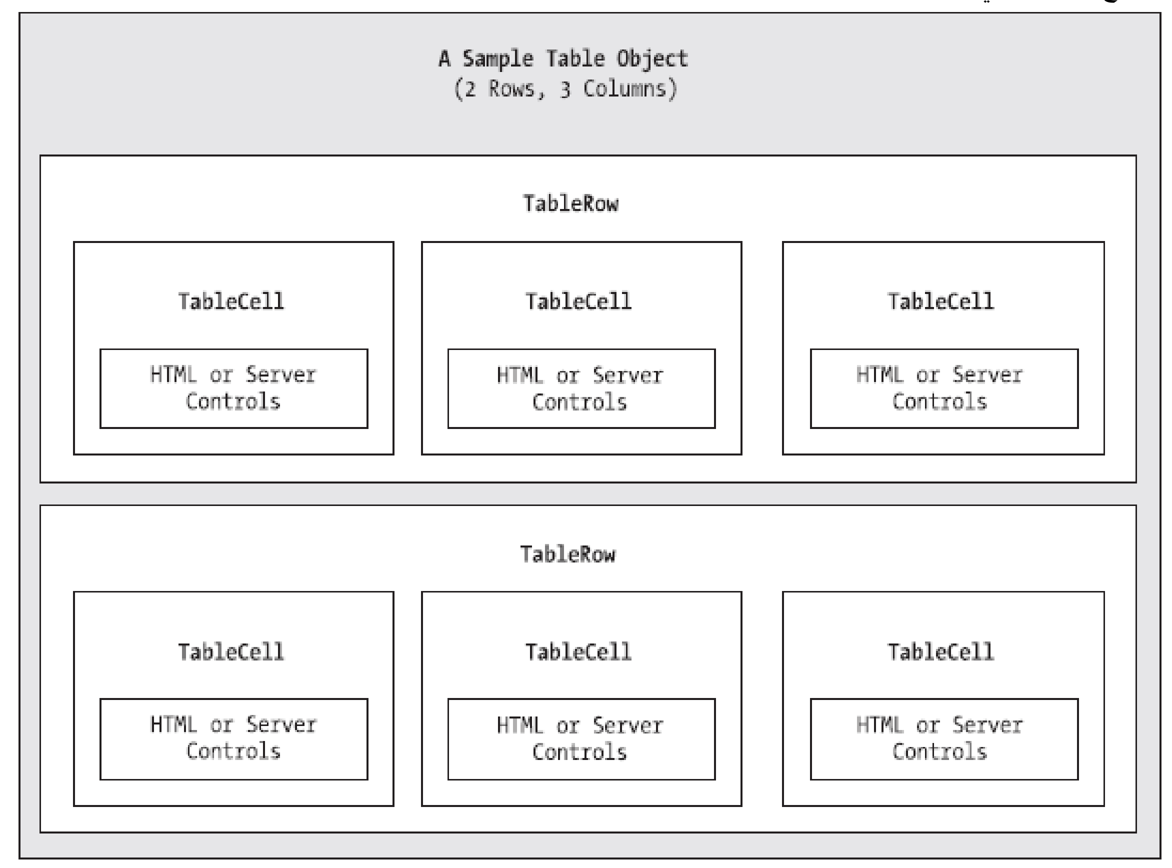


عناصر تحكم الجداول:

يتم بناء عناصر تحكم الجداول من هرمية من الأغراض حيث يحتوي كل غرض جدول غرض TableRow أو أكثر ويحتوي كل غرض TableRow بدوره، غرض TableCell أو أكثر.

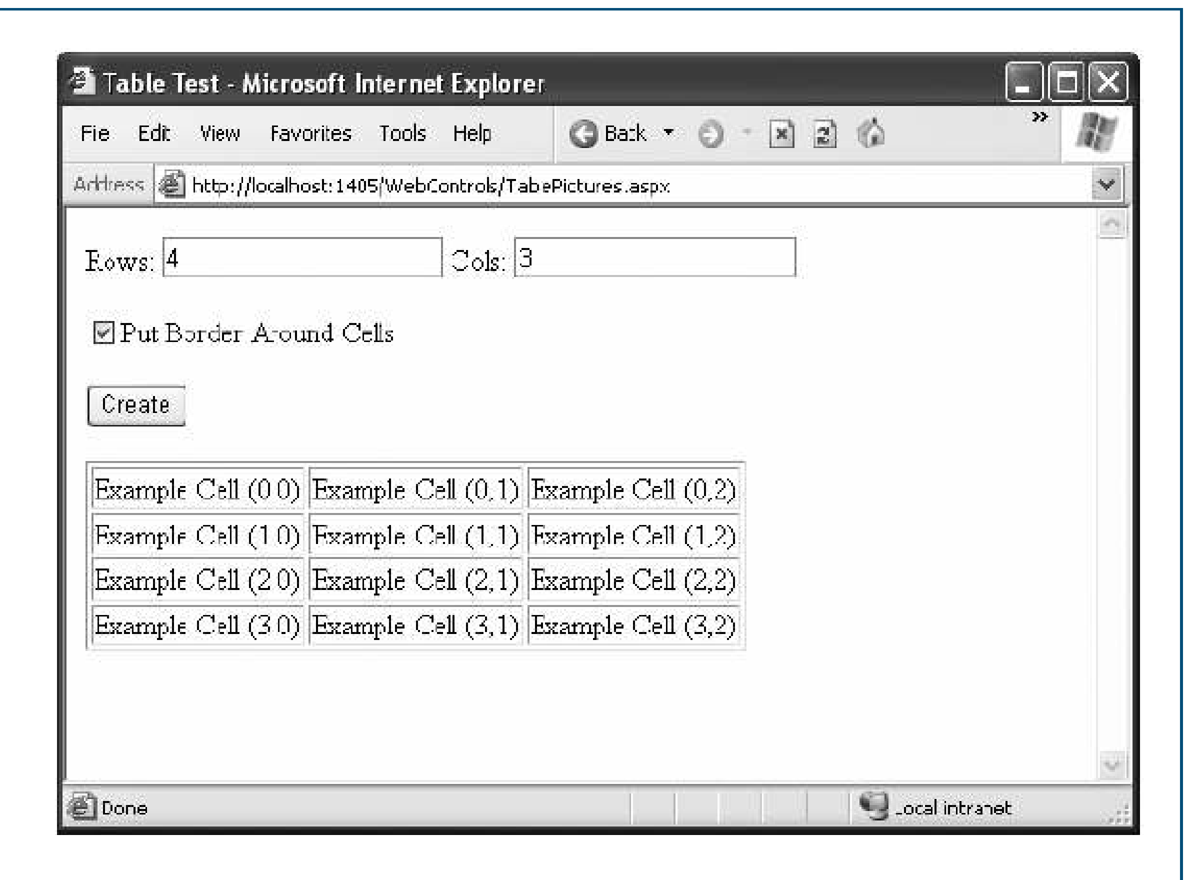
يحتوي كل غرض TableCell عناصر تحكم ASP.NET الأخرى التي تقوم بإظهار المعلومات ضمن هذه الخلية من الجدول.

يوضح الشكل التالي هذه الهرمية و علاقة الأغراض Tabel، TabelRow، TableCell بعضها ببعض:



مثال

يمكننا إتباع نفس الآلية لإنشاء جدول بشكل ديناميكي . الشكل التالي يمثل تطبيق بسيط. يستخدم هذا التطبيق القيم المدخلة عبر العلب النصية لتحديد عدد الصفوف و عدد الأعمدة للجدول المطلوب إنشاؤه.كما و يحدد عنصر تحكم CheckBox تفعيل أو إلغاء إحاطة الخلايا بإطار.



يوضح رماز صفحة aspx التالي كيفية إضافة عنصر تحكم Table عناصر تحكم TextBox و عنصر تحكم CheckBox إضافة إلى عنصر تحكم Button:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="TableTest.aspx.cs" Inherits="TableTest" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Table Test</title>
</head>
<body>
<form runat="server">
<div>
Rows:
<asp:TextBox ID="txtRows" runat="server" />
 Cols:
<asp:TextBox ID="txtCols" runat="server" />
<br /><br />
<asp:CheckBox ID="chkBorder" runat="server"
Text="Put Border Around Cells" />
<br /><br />
<asp:Button ID="cmdCreate" OnClick="cmdCreate_Click" runat="server"
Text="Create" />
<br /><br />
<asp:Table ID="tbl" runat="server" />
</div>
</form>
</body>
```

```
</html>
```

لا يحتوي عنصر تحكم الجدول Table أية صفوف أو خلايا عند إنشاءه، لذلك نحن بحاجة لتضمين تأشيرات أخرى ضمن تأشيرة عنصر التحكم Table لإظهار صفوف وخلايا.

يظهر المثال التالي جدول يحتوي صف واحد بخلية واحدة:

```
<asp:Table ID="tbl" runat="server">
<asp:TableRow ID="row" runat="server">
<asp:TableCell ID="cell" runat="server">A Sample Value</asp:TableCell>
</asp:TableRow>
</asp:Table>
```

لن نقوم باستخدام التأشيرات و بناء الجدول بشكل مباشر في مثالنا.

ستكون مهمة بناء محتوى الجدول ديناميكياً ملقاة على عاتق الرماز الخاص بمقبض حدث تحميل الصفحة و مقبض حدث ضغط الزر Create.

سيتم تحديد سماكة إطار الخلايا و نمط الإطار ضمن مقبض حدث تحميل الصفحة وتوليد الصفوف والخلايا ومحتواها ضمن مقبض حدث ضغط الزر Create.

```
public partial class TableTest : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
// Configure the table's appearance.
// This could also be performed in the .aspx file
// or in the cmdCreate_Click event handler.
tbl.BorderStyle = BorderStyle.Inset;
tbl.BorderWidth = Unit.Pixel(1);
}

protected void cmdCreate_Click(object sender, EventArgs e)
{
// Remove all the current rows and cells.
// This is not necessary if EnableViewState is set to false.
tbl.Controls.Clear();
int rows = Int32.Parse(txtRows.Text);
int cols = Int32.Parse(txtCols.Text);
for (int row = 0; row < rows; row++)
{
// Create a new TableRow object.
TableRow rowNew = new TableRow();
// Put the TableRow in the Table.
tbl.Controls.Add(rowNew);
for (int col = 0; col < cols; col++)
{
// Create a new TableCell object.
TableCell cellNew = new TableCell();
cellNew.Text = "Example Cell (" + row.ToString() + ",";
```

```
cellNew.Text += col.ToString() + ")";
if (chkBorder.Checked)
{
cellNew.BorderStyle = BorderStyle.Inset;
cellNew.BorderWidth = Unit.Pixel(1);
}
// Put the TableCell in the TableRow.
rowNew.Controls.Add(cellNew);
}
}
}
}
```

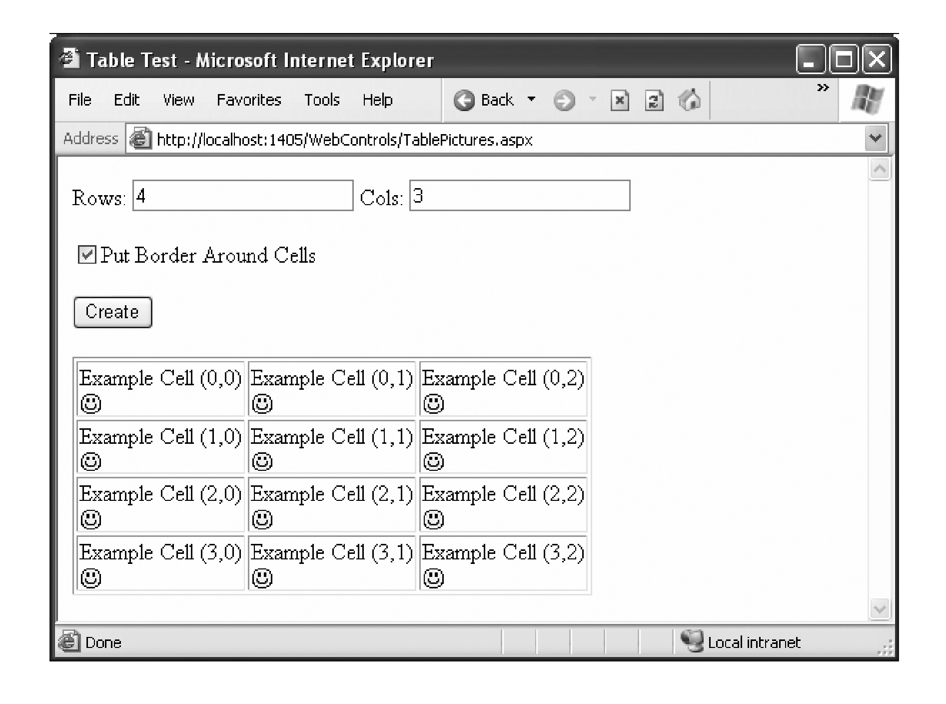
يستخدم النص السابق غرض المجموعة Controls لإضافة أغراض عناصر التحكم الأبناء. تتوفر الخاصة التي تعيد المجموعة Controls في جميع عناصر التحكم الحاوية.

يبين النص التالي كيفية الاستفادة من غرض المجموعة Controls لإضافة عناصر تحكم أخرى ضمن الخلايا مثل عنصر تحكم Image أو Label.

```
// Create a new TableCell object.
cellNew = new TableCell();
// Create a new Label object.
Label lblNew = new Label();
lblNew.Text = "Example Cell (" + row.ToString() + "," + col.ToString() +
")<br />";
System.Web.UI.WebControls.Image imgNew = new System.Web.UI.WebControls.Image();
imgNew.ImageUrl = "cellpic.png";
// Put the label and picture in the cell.
cellNew.Controls.Add(lblNew);
cellNew.Controls.Add(imgNew);
// Put the TableCell in the TableRow.
rowNew.Controls.Add(cellNew);
```

إن أهم ما يجعل عنصر تحكم Table مميزاً كون كل خلية ضمنه هي غرض متكامل المزايا. يمكن التعامل مع كل خلية بشكل منفصل و تحديد سماكة أو لون للإطار أو للنص داخلها و إضافة المحتوى المرغوب فيه.

يوضح الشكل التالي واجهة التطبيق بعد التعديل:



الأحداث الخاصة بعناصر تحكم الوب و عملية إعادة الإرسال الآلية AutoPostBack:

بينّا في الجلسة السابقة محدودية الأحداث التي تقدمها عناصر تحكم الوب ووضحنا أنها توفر حدثين فقط ServerClick لعناصر التحكم التي تتضمن إعادة الإرسال و الحدث ServerChange الخاص بالعناصر التي تقبل الإدخال مثل Input.

نعلم أن الرماز في ASP.NET يعالج على المخدم ثم يتم إرسال الخرج كصفحات HTML إلى الزبون.

يمكن معالجة الأحداث التي تسبب إعادة إرسال الصفحة مثل حدث Click. حيث يتم إطلاق الحدث و تنفيذ الرماز في مقبض الحدث من جهة المخدم.

لكن ماذا عن الأحداث التي لا تسبب إعادة إرسال الصفحة؟

كيف سيتم تفعيل الاستجابة للحدث؟

في الواقع هناك طريقتان :

1- الانتظار حتى عملية الإرسال التالية أي تجميع الأحداث لحين القيام بإرسال الصفحة .

مثلاً لن يحدث شيء بشكل فوري عند اختيار عنصر جديد من القائمة عند تعديل اختيار عنصر من قائمة SelectedIndexChanged و لكن عندما ينقر الزبون زر Submit لإرسال الصفحة يتم إطلاق حدثين الأول خاص بنقر الزر والثاني الخاص بالتعديل على العنصر المختار من القائمة.

2- الحل الآخر لهذه المشكلة هو استخدام الخيار AutoPostBack.

يجبر هذا الخيار عناصر التحكم على إعادة إرسال الصفحة فوراً عند تلقي حدث ما من جهة الزبون.

يتم بهذه الطريقة ارسال الصفحة و تنفيذ الرماز الخاص بمقبض الحدث على المخدم ثم يتم إعادة نسخة جديدة من الصفحة إلى الزبون تتضمن أية تعديلات سببها إطلاق الحدث.

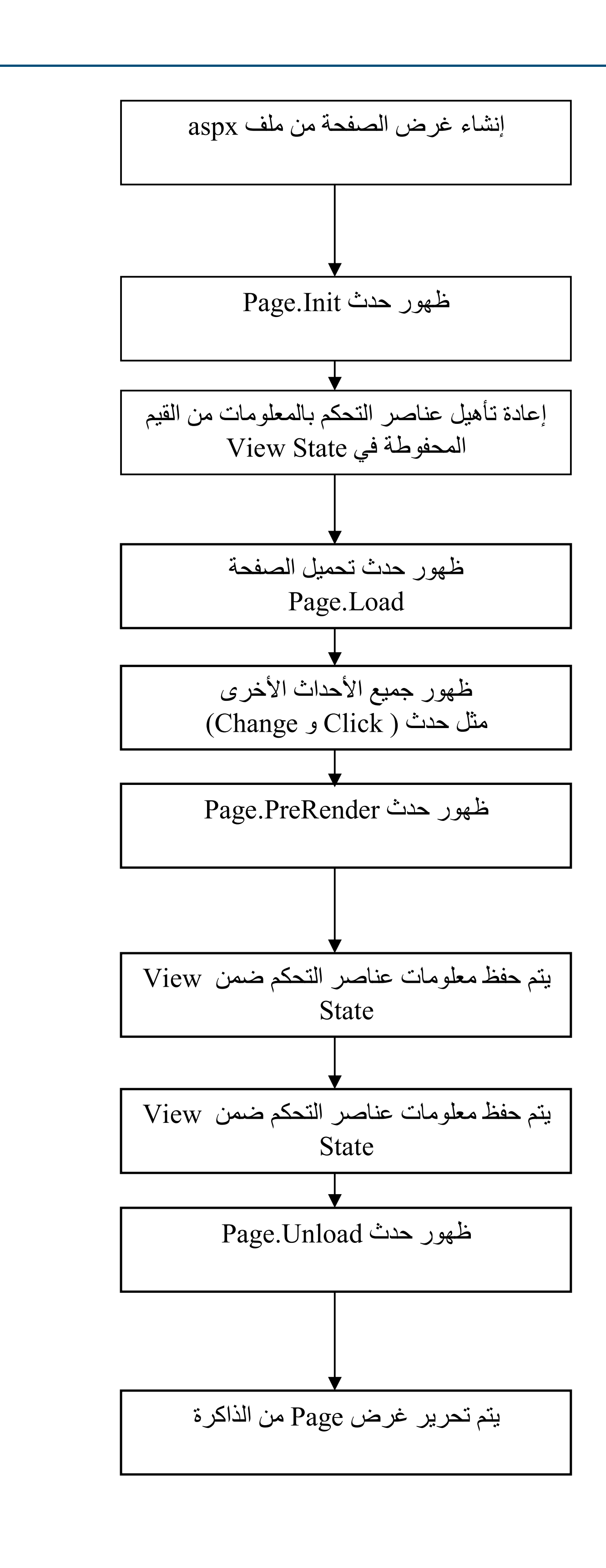
يؤدي استخدام الحل الذي يعتمد AutoPostBack في كثير من الأحيان إلى الحصول على صفحات أقل استجابة نظراً للتأخير المصحوب بعملية إعادة إرسال الصفحة مراراً و تكراراً عند الاستجابة لكل حدث (الحل في هذه الحالة قد يكون باستخدام مكونات AJAX).

تدعم جميع عناصر التحكم التي تقبل الإدخال الخاصة AutoPostBack.

يوضح الجدول التالي الأحداث الخاصة بعناصر تحكم وب الأساسية :

الحدث	عناصر تحكم الوب التي تدعم هذا الحدث	عملية إعادة الإرسال مفعلة بشكل دائم
Click	Button, ImageButton	True
TextChanged	TextBox	False
CheckedChanged	CheckBox, RadioButton	False
SelectedIndexChanged	DropDownList, ListBox, CheckBoxList, RadioButtonList	False

فيما يلي شكل يبين تسلسل معالجة عملية إعادة إرسال الصفحة بغرض تحديد المرحلة التي يتم عندها ظهور كل حدث من الأحداث:



إنشاء غرض الصفحة من ملف aspx
ظهور حدث Page.Init
إعادة تأهيل عناصر التحكم بالمعلومات من القيم المحفوظة في View State
ظهور حدث تحميل الصفحة Page.Load
ظهور جميع الأحداث الأخرى مثل حدث (Click و Change)
ظهور حدث Page.PreRender
يتم حفظ معلومات عناصر التحكم ضمن View State
يتم حفظ معلومات عناصر التحكم ضمن View State
ظهور حدث Page.Unload
يتم تحرير غرض Page من الذاكرة

كيف يعمل حدث Postback:

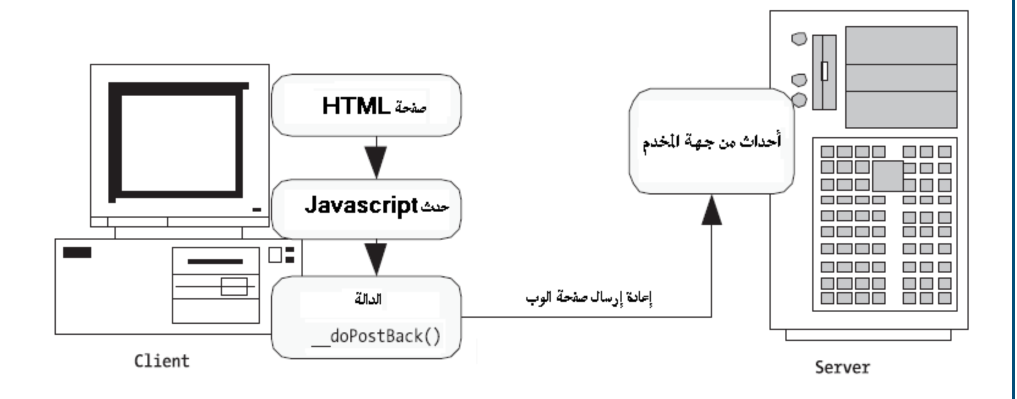
تقوم Asp.NET بتوليد رماز javascript من جهة الزبون و تنشأ دالة خاصة باسم doPostBack_ لتحسس الحدث (من جهة الزبون) و تأمين عملية إرسال الصفحة. الرماز التي تولد الدالة يشبه التالي:

```
<script language="text/javascript">
<!--
function __doPostBack(eventTarget, eventArgument) {
var theform = document.Form1;
theform.__EVENTTARGET.value = eventTarget;
theform.__EVENTARGUMENT.value = eventArgument;
theform.submit();
}
// -->
</script>
```

أما نص HTML الخاص بالعنصر فيكون شبيه بالتالي:

```
<select ID="lstBackColor" onchange="__doPostBack('lstBackColor','')" language="javascript">
```

يبين الشكل التالي كيف تحدث عملية إعادة الإرسال الآلي للصفحة:



دورة حياة الصفحة:

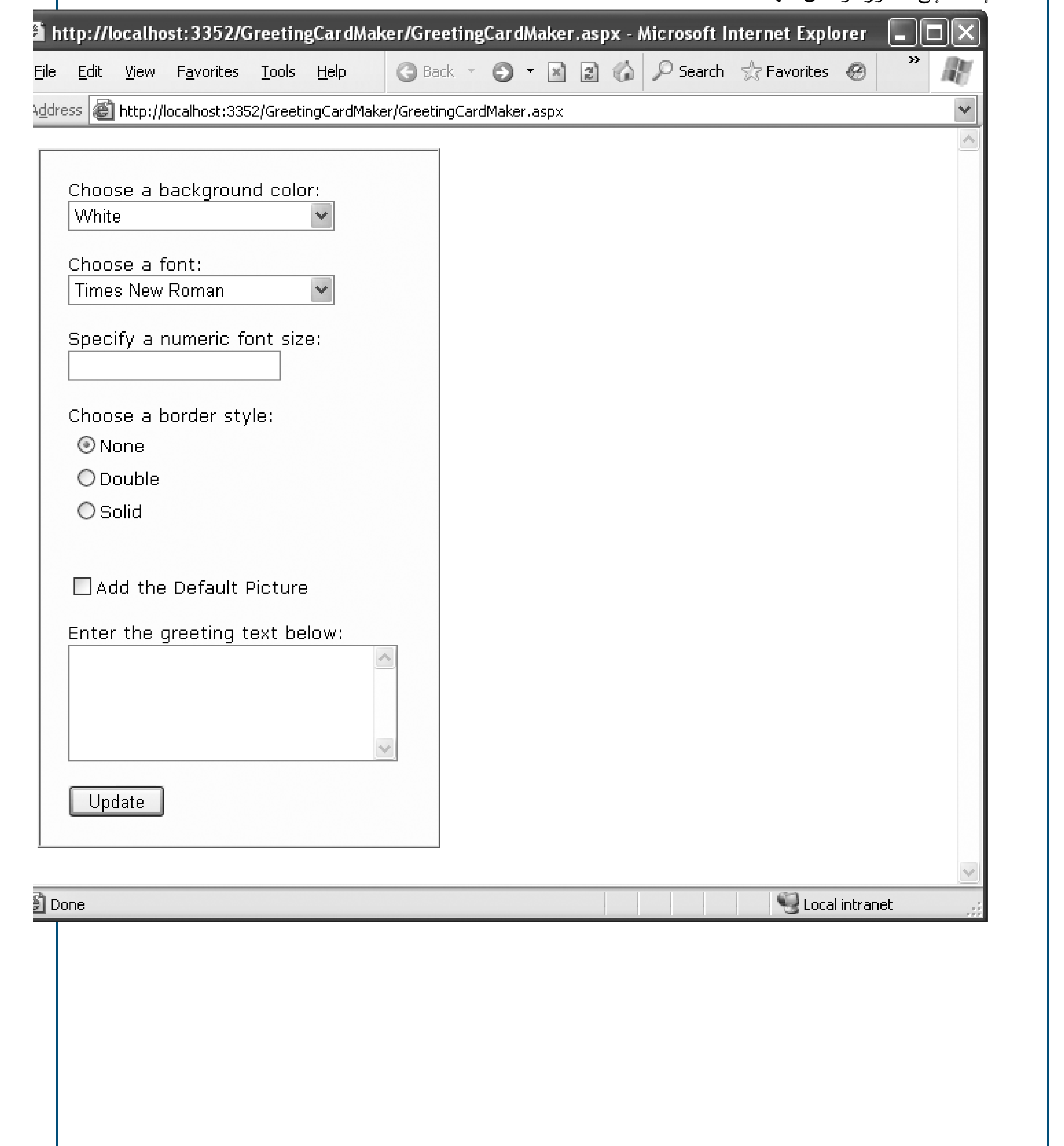
لنفهم أكثر كيف تعمل الأحداث في عناصر تحكم الوب لا بد لنا من فهم أعمق لدورة حياة الصفحة إذا افترضنا أنه تم تعيين الخاصة AutoPostBack إلى True ستكون المراحل التي تمر بها الصفحة هي التالية:

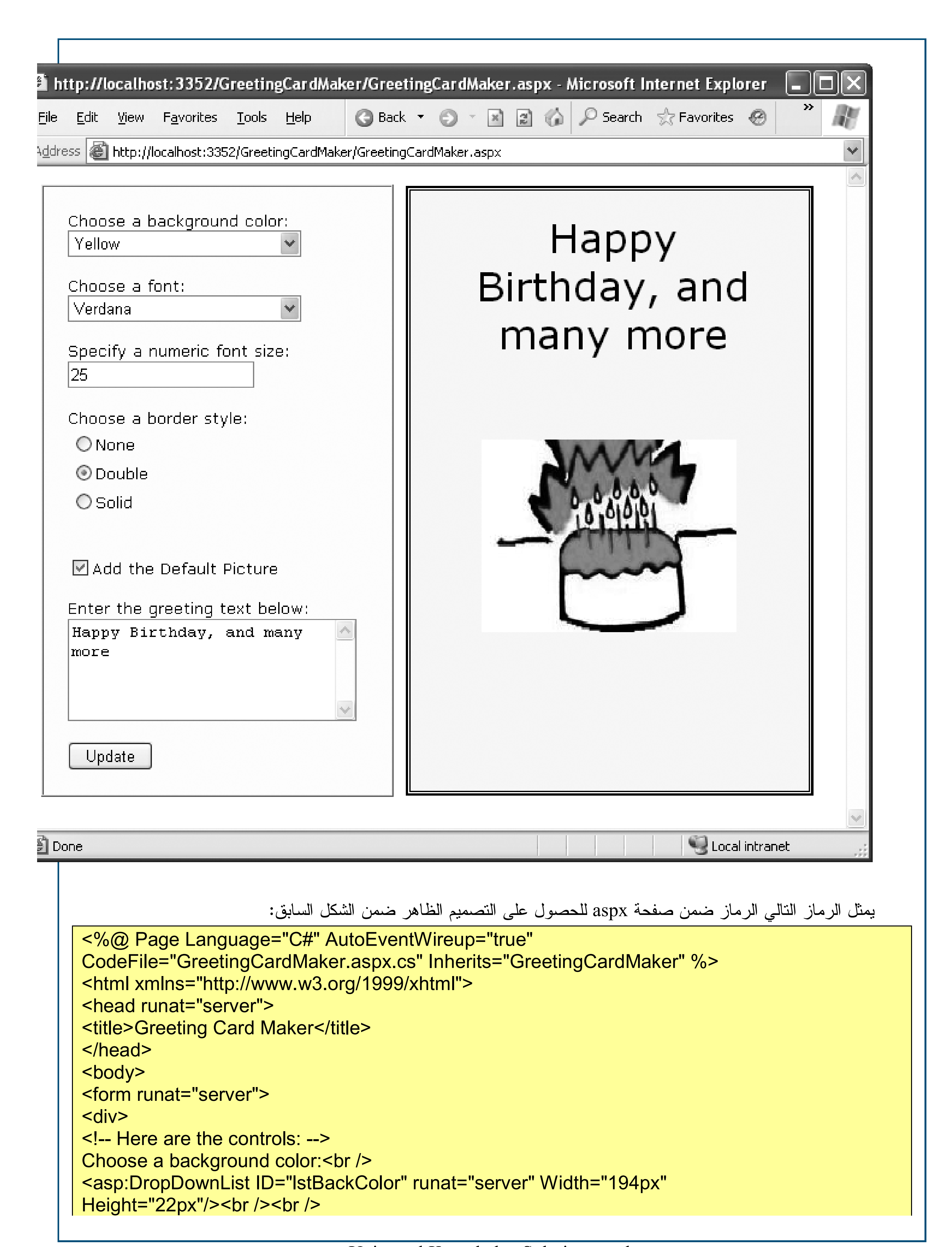
1. يتم تفعيل دالة dopostback javascript_من طرف الزبون و يتم إعادة إرسال الصفحة إلى المخدم.
2. تتم إعادة إنشاء غرض الصفحة باستخدام ملف aspx.
3. تقوم ASP.NET باستعادة معلومات الحالة من حقول View state المخفية و تحدث عناصر التحكم وفقاً لتلك المعلومات المخزنة.
4. يتم إطلاق حدث تحميل الصفحةPage.Load.
5. يتم إطلاق حدث Change المناسب لعنصر التحكم (ليس هناك ترتيب محدد لتنفيذ الأحداث في حال أكثر من عنصر تحكم).
6. يتم إطلاق حدث PageUnload و يتم تصيير الصفحة و تحويل مجموعة عناصر التحكم إلى صفحة HTML
7. يتم إرسال الصفحة الجديدة إلى الزبون.

مثال: تطبيق مصمم بطاقات تهنئة الكترونية

سنستخدم هذا المثال لنقوم بصقل مجموعة المعارف التي تم الحصول عليها خلال هذه الجلسة.

يوضح الشكل التالي الواجهة المراد الحصول عليها للتطبيق حيث يمكن اختيار لون الخلفية و نمط الخط و حجمه و نمط الإطار إضافة إلى الصورة و نص التهنئة.





يمثل الرماز التالي الرماز ضمن صفحة aspx للحصول على التصميم الظاهر ضمن الشكل السابق:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="GreetingCardMaker.aspx.cs" Inherits="GreetingCardMaker" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Greeting Card Maker</title>
</head>
<body>
<form runat="server">
<div>
<!-- Here are the controls: -->
Choose a background color:<br />
<asp:DropDownList ID="lstBackColor" runat="server" Width="194px"
Height="22px"/><br /><br />
```

```
Choose a font:<br />
<asp:DropDownList ID="lstFontName" runat="server" Width="194px"
Height="22px" /><br /><br />
Specify a numeric font size:<br />
<asp:TextBox ID="txtFontSize" runat="server" /><br /><br />
Choose a border style:<br />
<asp:RadioButtonList ID="lstBorder" runat="server" Width="177px"
Height="59px" /><br /><br />
<asp:CheckBox ID="chkPicture" runat="server"
Text="Add the Default Picture"></asp:CheckBox><br /><br />
Enter the greeting text below:<br />
<asp:TextBox ID="txtGreeting" runat="server" Width="240px" Height="85px"
TextMode="MultiLine" /><br /><br />
<asp:Button ID="cmdUpdate" OnClick="cmdUpdate_Click"
runat="server" Width="71px" Height="24px" Text="Update" />
</div>
<!-- Here is the card: -->
<asp:Panel ID="pnlCard" runat="server"
Width="339px" Height="481px"
HorizontalAlign="Center"><br /> 
<asp:Label ID="lblGreeting" runat="server" Width="256px"
Height="150px" /><br /><br /><br />
<asp:Image ID="imgDefault" runat="server" Width="212px"
Height="160px" />
</asp:Panel>
</form>
</body>
</html>
```

أما الرماز في الخلفية:

```
public partial class GreetingCardMaker : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
if (!this.IsPostBack)
{
// Set color options.
lstBackColor.Items.Add("White");
lstBackColor.Items.Add("Red");
lstBackColor.Items.Add("Green");
lstBackColor.Items.Add("Blue");
lstBackColor.Items.Add("Yellow");
// Set font options.
lstFontName.Items.Add("Times New Roman");
lstFontName.Items.Add("Arial");
lstFontName.Items.Add("Verdana");
lstFontName.Items.Add("Tahoma");
// Set border style options by adding a series of
// ListItem objects.
ListItem item = new ListItem();
// The item text indicates the name of the option.
```

```
item.Text = BorderStyle.None.ToString();
// The item value records the corresponding integer
// from the enumeration. To obtain this value, you
// must cast the enumeration value to an integer,
// and then convert the number to a string so it
// can be placed in the HTML page.
item.Value = ((int)BorderStyle.None).ToString();
// Add the item.
lstBorder.Items.Add(item);
// Now repeat the process for two other border styles.
item = new ListItem();
item.Text = BorderStyle.Double.ToString();
item.Value = ((int)BorderStyle.Double).ToString();
lstBorder.Items.Add(item);
item = new ListItem();
item.Text = BorderStyle.Solid.ToString();
item.Value = ((int)BorderStyle.Solid).ToString();
lstBorder.Items.Add(item);
// Select the first border option.
lstBorder.SelectedIndex = 0;
// Set the picture.
imgDefault.ImageUrl = "defaultpic.png";
}
}
protected void cmdUpdate_Click(object sender, EventArgs e)
{
// Update the color.
pnlCard.BackColor = Color.FromName(lstBackColor.SelectedItem.Text);
// Update the font.
lblGreeting.Font.Name = lstFontName.SelectedItem.Text;
if (Int32.Parse(txtFontSize.Text) > 0)
{
lblGreeting.Font.Size =
FontUnit.Point(Int32.Parse(txtFontSize.Text));
}
// Update the border style. This requires two conversion steps.
// First, the value of the list item is converted from a string
// into an integer. Next, the integer is converted to a value in
// the BorderStyle enumeration.
int borderValue = Int32.Parse(lstBorder.SelectedItem.Value);
pnlCard.BorderStyle = (BorderStyle)borderValue;
// Update the picture.
if (chkPicture.Checked)
{
imgDefault.Visible = true;
}
else
{
imgDefault.Visible = false;
}
// Set the text.
lblGreeting.Text = txtGreeting.Text;
```

```
}
}
```

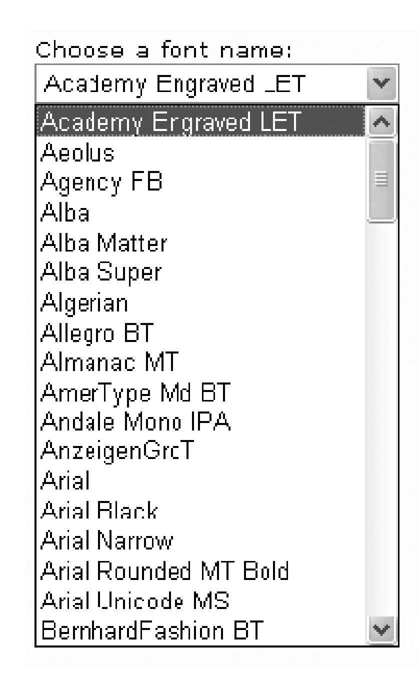
تحسين التطبيق:

نلاحظ في الرماز السابق أن قائمة الخطوط و الألوان تمت إضافتها يدوياً.

يمكن الاستحصال على معلومات الخطوط و الألوان المثبتة من خلال غرض المجموعة InstalledFontCollection كما يمكن استخدام الطريقة Enum.GetNames(typeof(KnownColor)) للحصول على قائمة بأسماء الألوان:

```
using System.Drawing;
using System.Drawing.Text;


// Get the list of available fonts, and add them to the font list.
InstalledFontCollection fonts = new InstalledFontCollection();
foreach (FontFamily family in fonts.Families)
{
lstFontName.Items.Add(family.Name);
}
```




```
// Get the list of colors.
string[] colorArray = Enum.GetNames(typeof(KnownColor));
lstBackColor.DataSource = colorArray;
lstBackColor.DataBind();
```

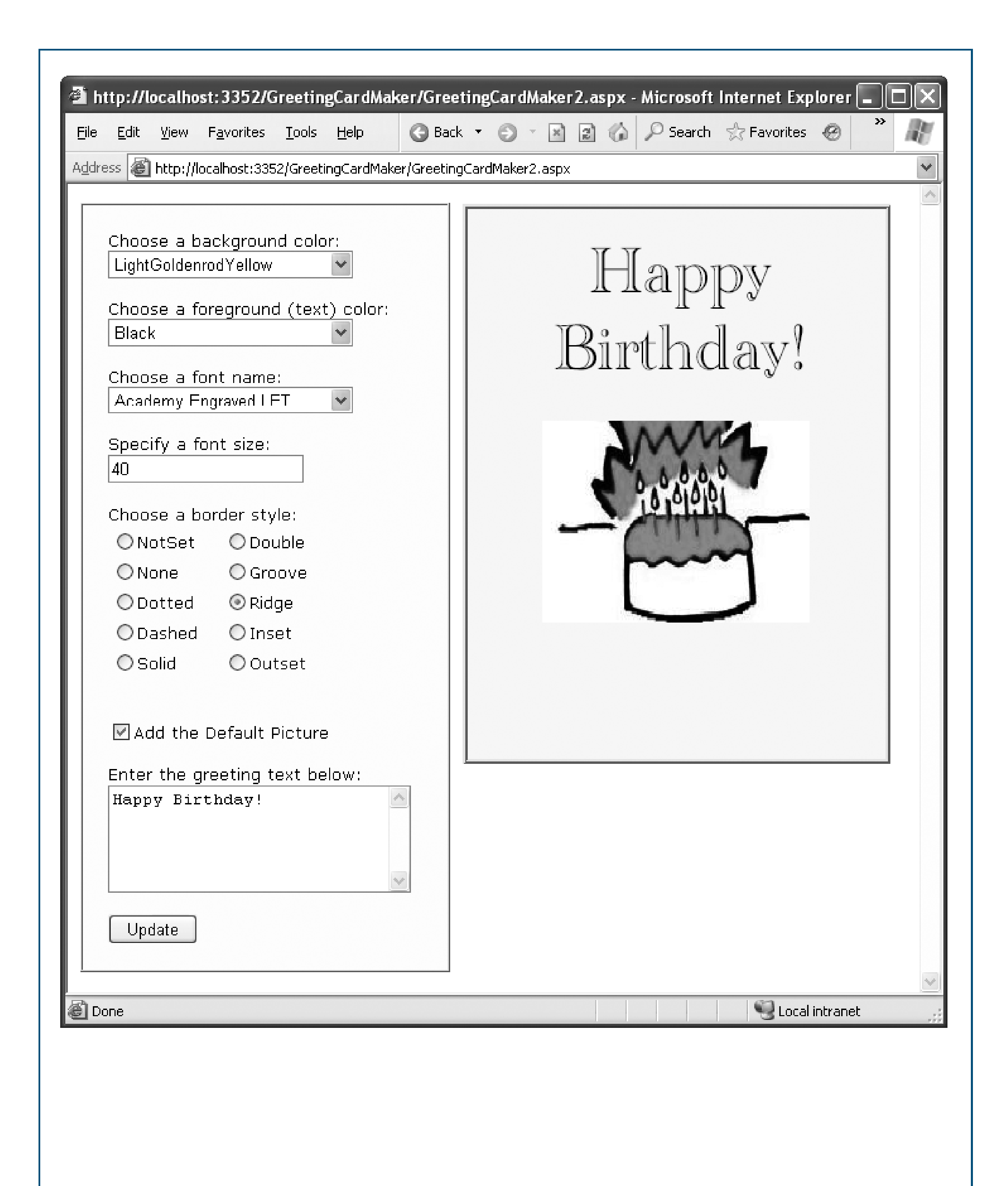
```
// Set border style options.
string[] borderStyleArray = Enum.GetNames(typeof(BorderStyle));
lstBorder.DataSource = borderStyleArray;
lstBorder.DataBind();
```

توليد البطاقة آلياً

ليتم توليد البطاقة آلياً سنقوم بإسناد القيمة True إلى الخاصة AutoPostBack وإضافة رماز تحديث البطاقة إلى مقبض حدث Click وحدث Change

```
Choose a background color:<br />
<asp:DropDownList ID="lstBackColor" AutoPostBack="true" runat="server"
Width="194px" Height="22px"/>
```

```
protected void ControlChanged(object sender, System.EventArgs e)
{
// Refresh the greeting card (because a control was changed).
UpdateCard();
}
protected void cmdUpdate_Click(object sender, EventArgs e)
{
// Refresh the greeting card (because the button was clicked).
UpdateCard();
}
private void UpdateCard()
{
// (The code that draws the greeting card goes here.)
}
```


http://localhost:3352/GreetingCardMaker/GreetingCardMaker2.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:3352/GreetingCardMaker/GreetingCardMaker2.aspx
Choose a background color:
LightGoldenrodYellow
Choose a foreground (text) color:
Black
Choose a font name:
Academy Engraved LET
Specify a font size:
40
Choose a border style:
NotSet
Double
None
Groove
Dotted
Ridge
Dashed
Inset
Solid
Outset
Add the Default Picture
Enter the greeting text below:
Happy Birthday!
Update
Happy Birthday!
Done
Local intranet

الفصل التاسع

إدارة الحالة

يتعرف الدارس في هذا الفصل على:

1. لماذا نحتاج إلى إدارة الحالة
2. إدارة الحالة ضمن صفحة وحيدة باستخدام المجموعة ViewStat
3. طرق نقل المعلومات بين الصفحات
4. استخدام الكعكات
5. استخدام سلسلة محارف الاستعلام
6. استخدام محددات URL المطعّمة
7. استخدام حالة الجلسة
8. حالة التطبيق

إدارة الحالة:

تعد إدارة الحالة من أهم عناصر الاختلاف بين البرمجة الخاصة بتطبيقات سطح المكتب و برمجة الوب و السؤال هو كيف سيتم تخزين المعلومات خلال فترة حياة التطبيق.

يمكن لهذه المعلومات أن تكون بسيطة كاسم المستخدم أو أكثر تعقيداً كالمحتوى الكامل لعربة تسوق ضمن تطبيق متجر بيع الكتروني. ففي حالة تطبيقات سطح المكتب لا داعي للقلق بشأن إدارة الحالة فالذاكرة متوفرة و لا حاجة للقلق إلا على مستخدم وحيد بينما تختلف القصة في تطبيقات الوب حيث يدخل آلاف المستخدمين بشكل متزامن لتشغيل التطبيق على نفس المخدم عبر اتصال HTTP الذي لا يمكن حفظ الحالة .هذه الظروف تجعل من المستحيل تصميم تطبيق بنفس الطريقة التقليدية المعتمدة في تطبيقات سطح المكتب.

في هذه الجلسة سوف نتعرف على كيفية استخدام مزايا إدارة الحالة في ASP.NET لتخزين المعلومات بعناية و باتساق. سنقوم باستعراض مجموعة من خيارات التخزين التي تتضمن ViewState ، غرض الجلسة ، الكعكات كذلك سنأخذ بعين الاعتبار كيفية انتقال المعلومات من صفحة إلى أخرى باستخدام الإرسال بين الصفحات و غرض QueryString.

مشكلة الحالة في تطبيقات الوب:

قد يبدو موقع ASP.NET احترافي و كأنه تطبيق مستمر العمل ، و لكن هذا الانطباع وهمي .ففي طلب الوب يتصل المستخدم بمخدم الوب و يطلب صفحة، عند تسليم الصفحة المطلوبة يتم إلغاء الاتصال و يقوم مخدم الوب بإهمال جميع المعلومات المتعلقة بالزبون. في الوقت الذي يستلم فيه الزبون الصفحة يكون البرنامج قد توقف عن العمل و لا يوجد أي معلومات باقية ضمن ذكرة مخدم الوب.

ViewState

المجموعة ViewState هي أحد الطرق الشائعة لتخزين المعلومات.

تستخدم ViewState حقول الحقول المخفية التي تقوم ASP.NET بإدراجها أوتوماتيكياً في ملف HTML الناتج بعد تصيير الصفحة.

تعتبر ViewState المكان الأنسب لتخزين المعلومات التي تستخدم في عمليات إعادة الإرسال المتكررة.
رأينا في الجلسات السابقة كيفية استخدام هذه الطريقة للحفاظ على المعلومات. مثلاً إذا تم تغيير قيمة نص سيتم الاحتفاظ بالمعلومات آلياً (التفعيل للخاصة EnableViewState تلقائي) باستخدام ViewState بهذه الطريقة يتم ضمان بقاء التعديلات التي تم إجراؤها على الصفحة و عدم عودة القيم إلى القيم البدئية التي تم تحديديها في زمن التصميم.
في جميع الأحوال لا يقتصر عمل ViewState على عناصر التحكم.إذ يمكن لرماز صفحة الوب إضافة معلومات بشكل مباشر إلى ViewState الخاصة بصفحة الوب و إستعادتها لاحقاً بعد إعادة إرسال الصفحة.
تتضمن أنماط البيانات التي يمكن تخزينها ضمن ViewState جميع أنماط البيانات من البسيطة و حتى الأغراض المخصصة التي ينشؤها المطور ضمن الصفحة.

غرض المجموعة ViewState:

توفر الخاصة ViewState المعلومات الحالية للصفحة. هذه الخاصة هي مثيل من صف المجموعة StateBag. إن صف StateBag هو مجموعة مفهرسة بمعنى أن كل عنصر ضمن المجموعة مخزن بشكل منفصل باستخدام اسم مميز من نمط سلسلة محرفية.
لنأخذ هذا الرماز على سبيل المثال :

```
// The this keyword refers to the current Page object. It's optional.
this.ViewState["Counter"] = 1;
```

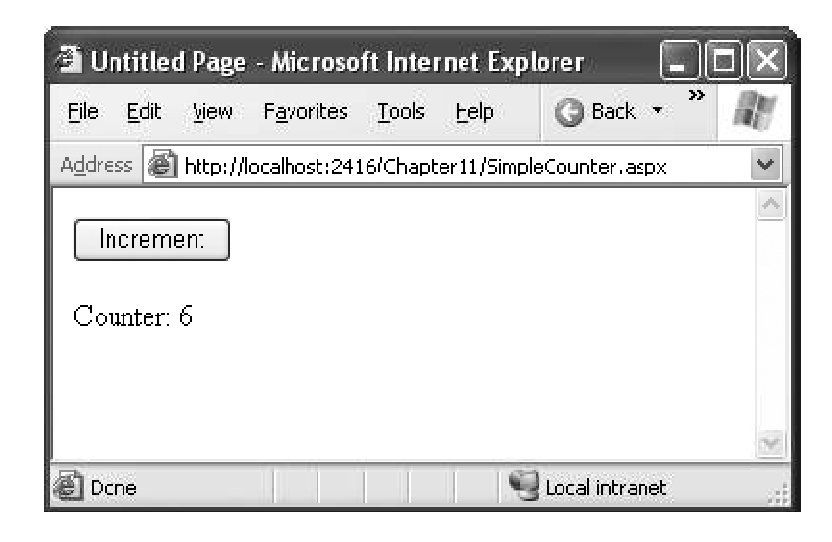
يقوم الرماز السابق بوضع القيمة 1 (عدد صحيح يحتوي الرقم 1) ضمن مجموعة ViewState ويقوم بإعطائها الاسم الوصفي "Counter".
إذا لم يكن هناك أي عنصر بالاسم "Counter" ستتم إضافة العنصر بشكل آلي.أما إذا كان هناك عنصر قد تم تخزينه تحت اسم "Counter" فسيتم استبداله.
عند استرجاع القيمة المخزنة نقوم باستخدام الاسم و لا بد أيضاً من تحويل القيمة المستعادة إلى نمط البيانات المطلوب باستخدام صيغ التحويل.
إن عملية التحويل مهمة لأن المجموعة ViewState تقوم بتخزين العناصر كأغراض أساسية مما يسمح لها بالتعامل مع عدة أنماط بيانات.
فيمايلي الرماز اللازم لاسترجاع قيمة العداد المخزنة تحت الاسم Counter من ViewState و تحويلها إلى عدد صحيح:

```
int counter;
counter = (int)this.ViewState["Counter"];
```

فيما يلي مثال يستخدم المجموعة ViewState ليقوم بتخزين عدد المرات التي تم ضغط زر فيها.
في هذا المثال عدم وجود أي صيغة لإدارة الحالة سوف تبقى قيمة العداد و بشكل دائم مساوية 1.

```
public partial class SimpleCounter : System.Web.UI.Page
{
protected void cmdIncrement_Click(Object sender, EventArgs e)
{
int counter;
if (ViewState["Counter"] == null)
{
counter = 1;
```

```
}
else
{
counter = (int)ViewState["Counter"] + 1;
}
ViewState["Counter"] = counter;
lblCount.Text = "Counter: " + counter.ToString();
}
}
```



جعل ViewState آمنة:

يتم تخزين معلومات ViewState كما ذكرنا ضمن حقول مخفية من الشكل:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="dDw3NDg2NTI5MDg7Oz4=" />
```

قد يخطر إلى أذهاننا أن القيمة غير المفهومة لViewState هي ناتج عملية. هذا غير صحيح، إذ أن معلومات ViewState يتم تحويلها إلى سلسلة محرفية للقاعدة 64 وهي نمط خاص من السلاسل المحرفية.

يعتمد هذا النمط في وثائق HTML كونه لا يتضمن أية محارف من المحارف الإضافية.

إذاً بإمكان أي مخترق و بعملية هندسة عكسية معرفة محتوى معلومات ViewState في ثواني.

لحماية المعلومات من التعديل يمكن الاعتماد على تفعيل خوارزميات التشويش و هي عملية إنشاء لحزمة إضافية من البيانات تم تشكيلها من تطبيق نمط من اختبار المجموع الحسابي و الاستعانة بمفتاح سري و إلحاق كتلة البيانات الناتجة بالقيمة الاساسية لـــ ViewState.

عند إعادة إرسال الصفحة يتم التأكد آلياً من التطابق بين محتوى ViewState ومطابقة كتلة التحقق المضافة مع الكتلة التي يتم إعادة توليدها باستخدام القيمة الحالية لViewState.

قد يلجأ بعض المطورون أحياناً لإزالة تفعيل هذه الخاصة وذلك عند التعامل مع مزرعة من المخدمات حيث لا يمكن التحقق محتوى ViewState لعدم توافر مفتاح التشويش السري الذي ولده أحد المخدمات عند المخدمات الأخرى.

لإزالة تفعيل التشويش يتم تعديل قيمة الواصفة enableViewStateMac ضمن ملف web.config أو machine.config:

```
<configuration>
<system.web>
<pages enableViewStateMac="false" />
...
</system.web>
</configuration>
```

كما يمكن حل هذه المشكلة بضبط المخدمات لتستخدم نفس مفتاح التشويش.

خصوصية معلومات ViewState:

حتى باستخدام مفتاح التشويش فإن محتوى البيانات سيظل مقروءاً من قبل المستخدم (وهذا أمر طبيعي في الحالة العامة) و لكن ماذا إذا احتوت المجموعة ViewState على معلومات لا بد من إبقائها غير معروفة فلا بد حينها من تعيين الواصفة ViewStateEncryptionMode إلى القيمة Always وذلك ضمن الرماز الخاص بملف web.config أو machine.config:

```
<%@Page ViewStateEncryptionMode="Always" %>
Or you can set the same attribute in a configuration file:
<configuration>
<system.web>
<pages viewStateEncryptionMode="Always" />
...</system.web></configuration>
```

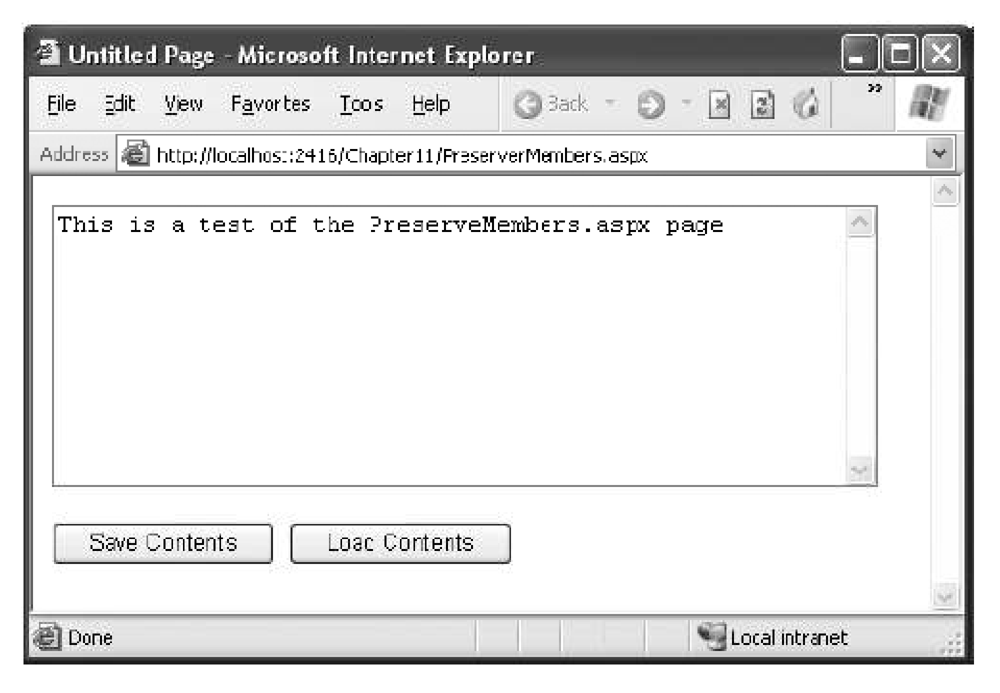
لدينا ثلاث خيارات بالنسبة لإعدادات التشفير الخاصة ب ViewState (Always، Never، Auto) وكما هو واضح من أسماء تلك الخيارات، الخيار Auto يحدد كون التشفير لن يحصل ما لم يطلب أحد العناصر على هذه الصفحة بشكل خاص.يقوم عنصر التحكم بطلب التشفير حال تلقيه الطريقة Page.RegisterRequiresViewStateEncryption(). أما في حال الخيار Never فلن تتم عملية تشفير محتوى المجموعة ViewState مما يوفر عبء هذه العملية.

المحافظة على متغيرات العناصر

للمحافظة على متغيرات العناصر يكفي حفظ هذه المتغيرات إلى الغرض ViewState ثم استعادته عند إعادة تحميل الصفحة.

المثال التالي يوضح هذه الفكرة من خلال استعادة قيمة نص علبة نصية تم حفظه.

يتم حفظ المعلومات المراد المحافظة عليها ضمن مقبض حدث Page.PreRender ثم إعادة تعيينها ضمن مقبض الحدث Page.Load




```
public partial class PreserveMembers : Page
{
// A member variable that will be cleared with every postback.
private string contents;
protected void Page_Load(Object sender, EventArgs e)
{
if (this.IsPostBack)
{
// Restore variables.
contents = (string)ViewState["contents"];
}
}

protected void Page_PreRender(Object sender, EventArgs e)
{
// Persist variables.
ViewState["contents"] = contents;
}

protected void cmdSave_Click(Object sender, EventArgs e)
{
// Transfer contents of text box to member variable.
contents = txtValue.Text;
txtValue.Text = "";
}

protected void cmdLoad_Click(Object sender, EventArgs e)
{
// Restore contents of member variable to text box.
txtValue.Text = contents;
}
}
```

يجب الانتباه في هذا السياق إلى عدم تخزين إلا ما كان ضرورياً من المعلومات باستخدام هذه التقنية. لأن هذا سيؤدي إلى ازدياد حجم الصفحة النهائية بشكل كبير و بالتالي إطالة زمن عملية انتقال الصفحة.

تخزين الأغراض المخصصة

يمكنك تخزين أغراضك المخصصة كما تقوم بتخزين المتحولات الصحيحة أو سلاسل المحارف ضمن غرض ViewState لكن في حالة الأغراض لا يمكن تخزينها ما لم يتم تحويلها إلى سياق من البايتات ليصبح من الممكن إرسالها ضمن الحقول المخفية التي تستخدمها ViewState لإرسال البيانات.

تسمى هذه العملية سلسلة و لا يمكن عادة بدونها وضع الأغراض التي تكون غير مسلسلة بطبيعتها ضمن غرض ViewState.

لجعل الأغراض قابلة للسلسلة يكفي إضافة الواصفة [Serializable] قبل تعريف الصف.

المثال التالي يستخدم هذه الطريقة لتخزين معلومات غرض Customer ضمن غرض ViewState.

```
[Serializable]
public class Customer
{
private string firstName;
public string FirstName
{
get { return firstName; }
set { firstName = value; }
}
private string lastName;
public string LastName
{
get { return lastName; }
set { lastName = value; }
}
public Customer(string firstName, string lastName)
{
FirstName = firstName;
LastName = lastName;
}
}
private string lastName;
public string LastName
{
get { return lastName; }
set { lastName = value; }
}
public Customer(string firstName, string lastName)
{
FirstName = firstName;
LastName = lastName;
```

```
}
}
```

بعد إضافة الواصفة Serializable أصبح بالإمكان تخزين المعلومات ضمن ViewState.

```
// Store a customer in view state.
Customer cust = new Customer("sami", "khiami");
ViewState["CurrentCustomer"] = cust;
```

عند استعادة معلومات الغرض المخزن لا بد من تحويلها قسرياً إلى نمط البيانات الخاص بالصف Customer:

```
// Retrieve a customer from view state.
Customer cust;
cust = (Customer)ViewState["CurrentCustomer"];
```

عند استخدام أغراض NET. يمكنك أن تعرف ما يمكن تخزينه ضمن ViewState بالاطلاع على ملف المساعدة الخاص بتعريف كل من هذه الأغراض فإذا وجدت الواصفة Serializable يمكنك التأكد بأن هذا الغرض يمكن تخزينه ضمن ViewState:

```
[Serializable]
[ComVisible(true)]
public sealed class FileInfo : FileSystemInfo
```

نقل المعلومات بين الصفحات

إن أحد أهم العناصر التي تحد من فائدة ViewState هي كونها مرتبطة بشكل كامل بصفحة محددة، وحين يتم طلب صفحة أخرى تزول كل المعلومات المخزنة في هذا الغرض.

تطرح ASP.NET لغرض نقل المعلومات بين الصفحات عدة حلول هي:

- استخدام تقنية إعادة الإرسال عبر الصفحات
- استخدام الغرض QueryString
- استخدام الكعكات
- استخدام أغراض الجلسة
- استخدام غرض التطبيق

استخدام تقنية إعادة الإرسال بين الصفحات:

إن عملية إعادة الإرسال بين الصفحات هي امتداد لآلية إعادة الإرسال التي قمنا باستخدامها.

يكمن الفارق في أن عملية الإرسال تتم إلى صفحة أخرى بشكل كامل متضمنة جميع المعلومات الخاصة بهذه الصفحة.

تبدو هذه التقنية بسيطة و لكنها، إذا لم تستخدم بعناية، قد تقود إلى سيناريو من مجموعة صفحات مرتبطة ببعضها بشكل كبير يصعب

تحسينها و تطويرها.

يتم دعم هذه الآلية باستخدام الخاصة PostBackUrl التي المعرفة في الواجهة البرمجية IButtonControl والتي توفرها عناصر التحكم ImageButton,LinkButtonKBUtton.

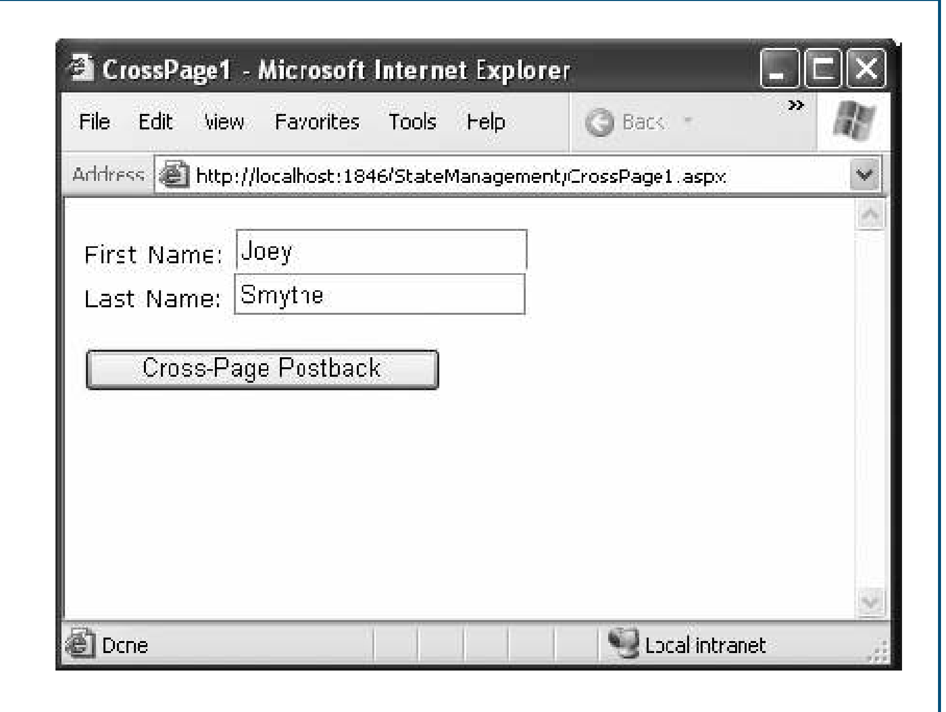
لاستخدام هذه التقنية يكفي اسناد قيمة المحدد الخاص بعنوان الصفحة المراد إرسال المعلومات إليها، إلى الخاصة PostBackUrl ضمن أحد عناصر التحكم السابقة.

عند النقر على الزر سيتم إعادة إرسال الصفحة مع القيم من جميع عناصر الإدخال إلى الصفحة الهدف.

المثال التالي يوضح عملية إرسال الصفحة CrossPage1.aspx الحاوية على نموذج يتضمن عنصر تحكم علبة نص وزر إلى الصفحة CrossPage2.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CrossPage1.aspx.cs"
Inherits="CrossPage1" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>CrossPage1</title>
</head>
<body>
<form id="form1" runat="server" >
<div>
First Name:
<asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>
<br />
Last Name:
<asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>
<br />
<br />
<asp:Button runat="server" ID="cmdPost"
PostBackUrl="CrossPage2.aspx" Text="Cross-Page Postback" /><br />
</div>
</form>
</body>
</html>
```

نلاحظ أن الصفحة CrossPage.aspx لا تحتوي أي رماز في الخلفية، فقط تم تعيين الواصفة PostBackUrl إلى القيمة CrossPage2.aspx



بعد أن يتم إرسال المعلومات إلى الصفحة الهدف تستطيع هذه الصفحة الوصول إلى معلومات الصفحة المرسلة عن طريق الغرض PreviousPage الذي مرجع إلى الصفحة السابقة:

```
public partial class CrossPage2 : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
if (PreviousPage != null)
{
lblInfo.Text = "You came from a page titled " +
PreviousPage.Title;
}
}
}
```

نلاحظ هنا بأننا قمنا بالتحقق من الصفحة السابقة لنضمن أن الطلب على الصفحة CrossPage2.aspx تم من خلال صفحة أخرى وليس مباشرة.

لكننا لن نتمكن من الاستفادة بشكل كامل من غرض الصفحة الذي قمنا باستعادته ما لم يتم تحويله قسرياً إلى نمط الصف المناسب وهو في حالتنا CrossPage1:

```
protected void Page_Load(object sender, EventArgs e)
{
CrossPage1 prevPage = PreviousPage as CrossPage1;
if (prevPage != null)
{
// (Read some information from the previous page.)
}
}
```

كما يمكن اللجوء إلى تعيين قيمة موجه PreviousPageType ضمن الصفحة الهدف إلى نمط البيانات المتوقع لهذا الغرض كما في

الرماز التالي:

```
<%@ PreviousPageType VirtualPath="~/CrossPage1.aspx" %>
```

ولكن حتى بعد اللجوء إلى أحد الطريقتين التاليتين لن تتمكن من الوصول إلى عناصر التحكم التي تحتويها الصفحة لأن هذه العناصر غير متاحة للوصول عبر الصفوف الأخرى.

لحل هذه المشكلة يمكنك إما إنشاء خصائص عامة تعيد مرجع للعناصر المطلوبة أو إنشاء طرق أو خصائص تعيد فقط ما يمكن أن تحتاجه الصفحات الأخرى من هذه الصفحة.

الرماز التالي يعرض هذه الفكرة:

```
public partial class CrossPage1 : System.Web.UI.Page
{
public string FullName
{
get { return txtFirstName.Text + " " + txtLastName.Text; }
}
}
```

وضمن رماز الصفحة CrossPage2.aspx يمكننا استخدام الخاصة التي قمنا بإنشائها:

```
protected void Page_Load(object sender, EventArgs e)
{
if (PreviousPage != null)
{
lblInfo.Text = "You came from a page titled " +
PreviousPage.Title + "<br />";
CrossPage1 prevPage = PreviousPage as CrossPage1;
if (prevPage != null)
{
lblInfo.Text += "You typed in this: " + prevPage.FullName;
}
}
}
```

ملاحظة: تعتبر طريقة إعدة الإرسال عبر الصفحات طريقة مفيدة و لكنها قد تقود إلى صفحات أكثر تعقيداً و بالأخص في الصفحات التي تستقبل طلبات من أكثر من صفحة.لذلك يفضل اعتماد هذه الطريقة في السيناريو الذي يتم تمرير المعلومات بين صفحتين محددتين فقط.

السلسلة المحرفية الخاصة بالاستعلام QueryString

من المقاربات الأخرى التي يمكن استخدامها عند الرغبة بإرسال بتمرير المعلومات بين الصفحات استعمال السلسلة المحرفية الخاصة بالاستعلام ضمن محدد المصدر الموحد URL.

تستخدم هذه المقاربة بشكل كبير ضمن محركات البحث، فمن المشاهد المألوفة رؤية عنوان ضمن حيز العنوان في المتصفح من الشكل:

```
http://www.google.ca/search?q=organic+gardening
```

ذلك الجزء من محدد URL اللاحق لإشارة الاستفهام هي QueryString . ففي المثال أعلاه يتم تعيين قيمة المتحول q إلى القيمة "organic+gradening".

الميزة الأساسية في هذه المقاربة التي تستخدمها QueryString أنها لا تشكل أي عبء على المخدم و لكن من وجهة تعتبر محدودة للعديد من الأسباب منها:

- المعلومات التي يمكن نقلها بين الصفحات بهذه الطريقة محصورة بالسلاسل المحرفية البسيطة التي لا بد أن تحتوي محارف URL النظامية.
- تظهر المعلومات التي يتم نقلها بشكل واضح للمستخدم و لأي شخص يتنصت على الانترنت.
- يمكن للمستخدم أن يقوم بتبديل المعلومات ضمن سلسة محارف الاستعلام يدوياً. ويقوم بإدخال معلومات مختلفة لا يقبلها البرنامج أو لا يستطيع حماية نفسه منها.
- تحدد العديد من المستعرضات طول السلسة المحرفية الخاصة بالاستعلام (عادة بين KB1 و 2KB). لهذا السبب، لا يمكن وضع كمية كبيرة من المعلومات ضمن سلسة محارف الاستعلام و التأكد من دعم جميع المستعرضات.

ولكن بالرغم من هذا تبقى طريقة إضافة المعلومات إلى سلسة محارف الاستعلام مفيدة جداً و بالأخص في التطبيقات المرتبطة بقواعد البيانات حيث يمكن ربط إظهار مجموعات مختلفة من النتائج من خلال تمرير معلومات تحدد السجلات المراد إعادتها. تستخدم العديد من تطبيقات التجارة الالكترونية هذه التقنية (فقط لاستعراض المنتجات و ليس للشراء).

لا بد من تخزين المعلومات ضمن سلسة محارف الاستعلام يدوياً إذ لا توفر ASP.NET مجموعة مخصصة لذلك أو يمكن استخدام الطريقة ()Response.Redirect أو عنصر تحكم HyperLink مخصص:

```
// Go to newpage.aspx. Submit a single query string argument
// named recordID, and set to 10.
Response.Redirect("newpage.aspx?recordID=10");
```

يمكننا إرسال أكثر من متحول مع قيمه بشرط فصلها بالإشارة & :

```
// Go to newpage.aspx. Submit two query string arguments:
// recordID (10) and mode (full).
Response.Redirect("newpage.aspx?recordID=10&mode=full");
```

الوصول إلى المعلومات التي تم إرسالها باستخدام سلسة محارف الاستعلام

لحسن الحظ، توفر ASP.NET مجموعة خاصة ضمن الغرض Request يمكن من خلالها الوصول إلى المعلومات التي تم إرسالها عبر سلسة محارف الاستعلام. اسم هذه المجموعة هو QueryString:

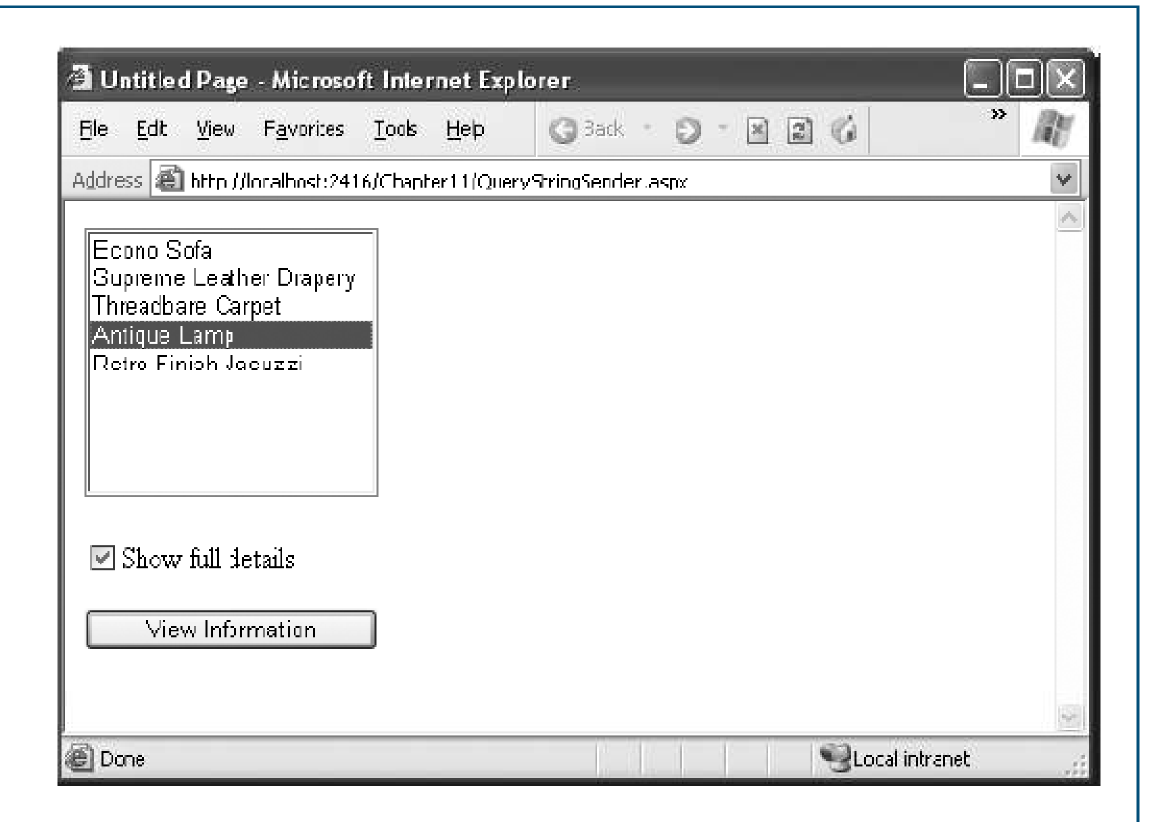
```
string ID = Request.QueryString["recordID"];
```

يتم استرجاع جميع قيم المتحولات كسلسلة محرفية و يقع على عاتق المطور تحويلها إلى نمط البيانات المطلوب.

تتم فهرسة القيم ضمن المجموعة QueryString باستخدام اسماء المتحولات المستعملة ضمن سلسة محارف الاستعلام.

مثال:

يقوم المثال التالي باستخدام مقاربة سلسلة محارف الاستعلام لإظهار المعلومات المرتبطة بدليل العنصر الذي تم اختياره من عنصر تحكم قائمة في أحد الصفحات ضمن صفحة ثانية.



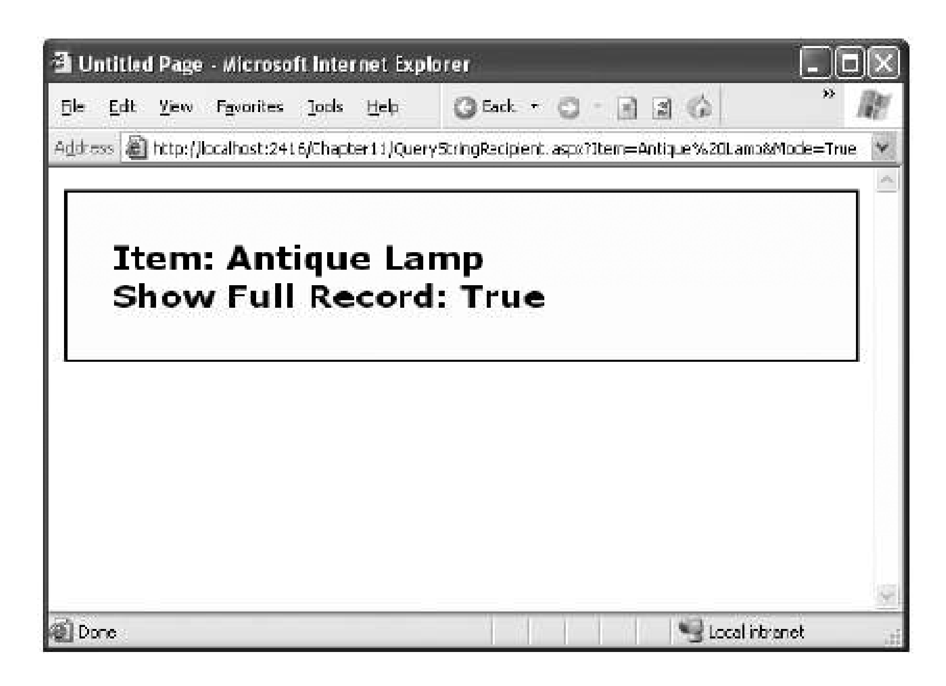
أما النص البرمجي للصفحة المرسلة فهو:

```
public partial class QueryStringSender : System.Web.UI.Page
{
protected void Page_Load(Object sender, EventArgs e)
{
if (!this.IsPostBack)
{
// Add sample values.
lstItems.Items.Add("Econo Sofa");
lstItems.Items.Add("Supreme Leather Drapery");
lstItems.Items.Add("Threadbare Carpet");
lstItems.Items.Add("Antique Lamp");
lstItems.Items.Add("Retro-Finish Jacuzzi");
}
}
protected void cmdGo_Click(Object sender, EventArgs e)
{
if (lstItems.SelectedIndex == -1)
{
lblError.Text = "You must select an item.";
}
else
{
// Forward the user to the information page,
// with the query string data.
string url = "QueryStringRecipient.aspx?";
url += "Item=" + lstItems.SelectedItem.Text + "&";
url += "Mode=" + chkDetails.Checked.ToString();
```

```
Response.Redirect(url);
}
}
}
```

والنص البرمجي في الصفحة المستقبلة فهو كما يلي:

```
public partial class QueryStringRecipient : System.Web.UI.Page
{
protected void Page_Load(Object sender, EventArgs e)
{
lblInfo.Text = "Item: " + Request.QueryString["Item"];
lblInfo.Text += "<br />Show Full Record: ";
lblInfo.Text += Request.QueryString["Mode"];
}
}
```



ترميز URL:

مشكلة المحارف غير المسموحة هي أحد المشاكل التي ترتبط باستخدام مقاربة سلسلة محارف الاستعلام.

يجب أن تكون جميع المحارف المستخدمة أحرف أو أرقام أو أحد المحارف الخاصة بما يتضمن ($ − _ . + ! * . ( )).

تقبل بعض المستعرضات (internet Explorer) محارف إضافية أخرى.

هذه المحارف غير المدعومة من العديد من المستعرضات مما يجعل استعمالها غير مأمون.

بعض المحارف لها معنى خاص مثل إشارة & التي تستخدم كما ذكرنا لفصل قيم المتحولات. كذلك الأمر بالنسبة لإشارة (+) التي تعتبر بديلاً عن استخدام محرف الفراغ. و إشارة (#) التي تستخدم لتحديد إشارة مرجعية ضمن صفحة الوب.

إذا أردنا إرسال بيانات تحوي هذه المحارف، سيحصل نقص في البيانات ناتج عن ترجمة هذه المحارف بشكل مختلف.

لتفادي هذه المشكلة نلجأ إلى تشفير النصوص قبل وضعها ضمن URL.

نستخدم لهذا الغرض تشفير خاص يقوم بتحويل المحارف الخاصة إلى نمط مقبول ضمن محدد URL.

يعتمد هذا التشفير على استبدال المحارف الخاصة بإشارة % متبوعة برقم من خانتين.

مثلاً يمكن تشفير المحرف & بشكل 26%

الاستثناء الوحيد لهذه القاعدة هو محرف الفراغ التي يمكن استبداله بـــ 20% أو بإشارة +.

تستخدم لغرض عملية التشفير الطريقة UrlEncode() في حين يتم استخدام الطريقة UrlDecode() لفك التشفير.

تتبع الطرق السابقة الصف HttpServerUtitlity .

يوضح المثال التالي استخدام هذه الطرق (UrlDecode،UrlEncode):

```
string url = "QueryStringRecipient.aspx?";
url += "Item=" + Server.UrlEncode(lstItems.SelectedItem.Text) + "&";
url += "Mode=" _ chkDetails.Checked.ToString();
Response.Redirect(url);
```

نلاحظ في هذا المثال السابق أننا لم نقم بتشفير المحرف & لأننا نريد منه أن يعمل كمحرف خاص يدمج المتحولين معاً.

تقوم ASP.NET تلقائياً بفك تشفير السلسلة في حال الوصول إليها باستخدام الغرض Request باستخدام المجموعة QueryString لذلك و في الحالة السابقة لا داعي لاستخدام عملية فك تشفير باستخدام الطريقة UrlDecodeعلى السلسلة التي تم استقبالها.

الكعكات

يعد استخدام الكعكات أحد الطرق الأخرى الشائعة لحفظ الحالة.

تساعد الكعكات على تخزين المعلومات ثم إعادة استخدامها.

الكعكات هي عبارة عن ملفات صغيرة يتم إنشاؤها على القرص الصلب للزبون (أو ضمن الذاكرة المؤقتة لمستعرض الوب).

من أهم الميزات في الكعكات:

- كونها تعمل بشكل شفاف دون علم المستخدم بالمعلومات التي تحتاج إلى تخزين
- إمكانية استخدامها من قبل أي صفحة ضمن التطبيق
- الاحتفاظ بها يتم لفترة أطول مما يسمح لاستخدامها في تخزين المعلومات بين الزيارات بمدى أطول

أما عيوب هذه الطريقة فهي تماثل تلك المتعلقة باستخدام سلسلة محارف الاستعلام بمعنى:

- إمكانية التعامل مع سلاسل المحارف البسيطة فقط
- يمكن الوصول إليها و قراءتها من قبل المستخدم

كل ما سبق يجعل خيار استخدام الكعكات خيار غير مناسب للمعلومات المعقدة والتي تعتبر الخصوصية من الأولويات أو تحتاج إلى تخزين كم كبير من المعلومات.

يقوم بعض المستخدمين بإلغاء دعم الكعكات في المستعرضات التي يستخدمونها أو يقومون بشكل يدوي بمسح ملفات الكعكات مما يخلق مشكلة للتطبيقات التي تستخدم هذه المقاربة.

لاستخدام الكعكات لا بد من استيراد فضاء الأسماء System.Net.

يعتبر التعامل مع الكعكات عملية سهلة حيث يوفر كل من غرضي Request و Response مجموعة باسم Cookies.

يتم اسناد قيمة إلى الكعكة باستخدام الغرض Response و استعادة القيمة بواسطة الغرض Request.

لتعيين قيمة كعكة:

- نقوم بإنشاء غرض HttpCookie
- نقوم بتأهيل هذا الغرض بالمعلومات المطلوب تخزينها على شكل ثنائيات اسم و قيمة كما يظهر في الرماز التالي:

```
// Create the cookie object.
HttpCookie cookie = new HttpCookie("Preferences");
// Set a value in it.
cookie["LanguagePref"] = "English";
// Add another value.
cookie["Country"] = "US";
// Add it to the current web response.
Response.Cookies.Add(cookie);
```

إذا تم إنشاء كعكة ما بهذه الطريقة تظل قيمة الكعكة موجودة لحين إغلاق المستعرض.

خلال ذلك الوقت يتم إرسال هذه الكعكة مع كل طلب على الوب.

لتحديد عمر الكعكة تستخدم الخاصة Expires كما في الرماز التالي:

```
// This cookie lives for one year.
cookie.Expires = DateTime.Now.AddYears(1);
```

يمكننا استعاد قيمة الكعكات من خلال غرض المجموعة Requested.Cookies:

```
HttpCookie cookie = Request.Cookies["Preferences"];
// Check to see whether a cookie was found with this name.
// This is a good precaution to take,
// because the user could disable cookies,
// in which case the cookie will not exist.
string language;
if (cookie != null)
{
language = cookie["LanguagePref"];
}
```

لحذف كعكة نقوم باستبدالها بكعكة أخرى تمتلك تاريخ انتهاء سابق.

الرماز التالي يوضح هذه التقنية:

```
HttpCookie cookie = new HttpCookie("LanguagePref");
cookie.Expires = DateTime.Now.AddDays(-1);
Response.Cookies.Add(cookie);
```

مثال على استخدام الكعكات

في المثال التالي سيتم استخدام الكعكات لتخزين اسماء الزبائن و إظهار رسالة ترحيب في حال وجد هذا الاسم مسبقا.



يكون رماز هذه الصفحة من الشكل:

```
public partial class CookieExample : System.Web.UI.Page
{
protected void Page_Load(Object sender, EventArgs e)
{
HttpCookie cookie = Request.Cookies["Preferences"];
if (cookie == null)
{
lblWelcome.Text = "<b>Unknown Customer</b>";
}
else
{
lblWelcome.Text = "<b>Cookie Found.</b><br /><br />";
lblWelcome.Text += "Welcome, " + cookie["Name"];
}
}
protected void cmdStore_Click(Object sender, EventArgs e)
{
// Check for a cookie, and only create a new one if
// one doesn't already exist.
HttpCookie cookie = Request.Cookies["Preferences"];
```

```
if (cookie == null)
{
cookie = new HttpCookie("Preferences");
}
cookie["Name"] = txtName.Text;
cookie.Expires = DateTime.Now.AddYears(1);
Response.Cookies.Add(cookie);
lblWelcome.Text = "<b>Cookie Created.</b><br /><br />";
lblWelcome.Text += "New Customer: " + cookie["Name"];
}
}
```

حالة الجلسة

تصل معظم التطبيقات إلى مرحلة تصبح فيها احتياجات التخزين أكثر تعقيداً.

وفي العديد من الحالات لا يمكن استخدام الكعكات أو إرسال المعلومات عبر QueryString و بالأخص عند الحاجة إلى تخزين الأغراض المخصصة أو الوصول إلى عناصر قواعد البيانات أو في الحالات التي تفرض فيها طبيعة التطبيق الحاجة إلى درجة أمان عالية.

استخدام غرض حالة الجلسة هو الحال الأمثل في هذه الحالة.

تعد إدارة الحالة باستخدام غرض الجلسة أحد أهم المزايا في ASP.NET للأسباب التالية:

- توفر هذه التقنية إمكانية تخزين أي نمط من البيانات على ذاكرة المخدم
- تكون البيانات المخزنة ضمن غرض الجلسة محمية لأنه لا يتم نقلها إلى الزبون
- ترتبط معلومات غرض الجلسة بجلسة واحدة محددة
- يتم تخصيص غرض جلسة مستقل لكل زبون
- يمكن للزبون تخزين بياناته ضمن غرض الجلسة

من الامثلة الشائعة لإدارة حالة الجلسة هو تطبيق سلة التسوق حيث يمكن للمستخدم تخزين معلومات المشتريات أثناء التسوق والتجوال بين أكثر من صفحة.

تتبع وملاحقة الجلسات:

تستخدم Asp.NET معرف فريد بطول 120 بت لملاحقة الجلسات.

يتم توليد أرقام هذه الجلسات بخوارزمية خاصة.

تضمن هذه الخوارزمية عشوائية كافية تمنع أي محاولة لتخمين أواستخدام تقنيات الهندسة العكسية لتحديد معرف الجلسة لزبون ما.

لذلك فالجزء الوحيد من معلومات الجلسة الذي يتم تبادله بين المخدم و الزبون هو ذلك الخاص بالمعرف الفريد لهذه الجلسة.

عندما يرسل الزبون معرف الجلسة تقوم Asp.NET بالبحث عن الجلسة الموافقة.

تقوم Asp.NET بوضع المعلومات التي تم الاستحصال عليها باستخدام معرف الجلسة تلقائياً ضمن مجموعة خاصة ليتم من خلالها

الوصول إلى المعلومات المخزنة.

يعتمد هذا السيناريو كما ذكرنا على أن يزود يتقدم الزبون بمعرف جلسة صالح مع كل طلب. يمكن تحقيق هذا الغرض بطريقتين:

1. استخدام الكعكات: في هذه الحالة يتم إرسال معرف الجلسة ضمن كعكة خاصة باسم (ASP.NET_SessionId). تقوم Asp.NET بإنشاء هذه الجلسة آلياً عندما يتم استخدام المجموعة Session الخاصة بإدارة الحالة. تعد هذه الطريقة هي الطريقة الافتراضية في العمل.
2. استخدام محددات URL المعدلة: يتم في هذه الحالة نقل معرف الجلسة من خلال محددات URL معدلة (تسمى مطعّمة أحياناً). تمكن هذه الطريقة من إدارة حالة الجلسة ضمن بيئة الزبون التي لا تدعم الكعكات.

لا تأتي الميزات والمرونة التي تقدمها حالة الجلسة بشكل مجاني. فبالرغم من كونها توفر تقنية تتجاوز العديد من نقاط الضعف التي تعاني منها الطرق الأخرى.فهي تجبر المخدم على تخزين معلومات إضافية في الذاكرة.

رغم كون المعلومات التي تخزن عادة ضمن غرض الجلسة صغيرة لكنها تصبح عبء حقيقي يؤثر على الأداء بشكل كبير عندما يرتفع عدد الزبائن الذين يحاولون الوصول إلى الموقع.

استخدام حالة الجلسة:

يمكن التعامل مع حالة الجلسة من خلال الصف Web.SessionState.HtpSessionState.

يتوفر هذا الصف من خلال الغرض المبّيت ضمن صفحة الوب Session.

تستخدم إدارة حالة الجلسة صيغة مشابهة لتلك التي قمنا باستخدامها مع غرض ViewState.

```
Session["InfoDataSet"] = dsInfo;
```

لاسترجاع المعلومات نستخدم الصيغة التالية، مع مراعاة إجراء التحويل إلى النمط المناسب:

```
dsInfo = (DataSet)Session["InfoDataSet"];
```

تعتبر حالة الجلسة عامة لكامل التطبيق لكن لمستخدم وحيد.

تحذف معلومات الجلسة في مجموعة من الحالات منها:

- إذا قام المستخدم بإغلاق وإعادة تشغيل المستعرض
- إذا قام المستخدم بالوصول إلى التطبيق باستخدام نافذة أخرى للمستعرض سيتم التعامل مع هذا الوصول كمستخدم جديد بمعرف مختلف
- إذا انقضى زمن محدد دون أي تفاعل مع التطبيق من قبل مستخدم محدد.يتم تحديد زمن انتهاء الصلاحية بتحديد قيمة الخاصة timeout و التي سنأتي على شرحها لاحقاً
- إذا قام الرماز بإنهاء حياة الجلسة قسرياً باستخدام الطريقة Abondon

تبقى الجلسة في الحالتين الأولى والثانية ضمن الذاكرة على المخدم لأن المخدم لا يملك أي فكرة عن كون الزبون قد أغلق الجلسة أو فتح نافذة أخرى. ستتجمع أغراض Session هذه ضمن الذاكرة بانتظار انتهاء صلاحيتها.

الصف HttpSessionState

الجدول التالي يوضح أهم الخصائص والطرق الخاصة بالصف HttpSessionState:

العنصر	الوصف
Count	تعيد هذه الخاصة عدد العناصر ضمن مجموعة ضمن غرض المجموعة الخاص بالجلسة.
IsCookieless	تحدد آلية تتبع معرف الجلسة سواء باستخدام الكعكات أو محددات URL المعدلة. في حال إسناد القيمة false لهذه الخاصة سيتم استخدام الكعكات.
IsNewSession	في حال إسناد القيمة true لهذه الخاصة سوف يتم التعامل مع أي جلسة في التطبيق على أنها جلسة جديدة ة لن تحاول Asp.NET متابعتها بل سيتم إنشاء جلسة جديدة مع كل طلب.
Mode	توفر قيمة مرقمة تحدد كيف ستخزن ASP.NET معلومات حالة الجلسة.يتم تحديد نمط التخزين هذا بناء على الإعدادات المعينة ضمن ملف web.config
SessionID	تعيد هذه الخاصة سلسة محرفية تتضمن معرف الجلسة للزبون الحالي.
Timeout	تحدد هذه الخاصة الزمن المنقضي بالدقائق قبل حذف حالة الجلسة في حال لم يتم تلقي أي طلب من الزبون خلال هذه الفترة. يمكن تغيير قيمة هذه الخاصة برمجياُ لتمكين إطالة أة إنقاص عمر حالة الجلسة حسب الحاجة.
Abandon()	تقوم هذه الطريقة بإلغاء الجلسة الحالية بشكل فوري و تحرير الذاكرة التي تشغلها. يفضل استخدام هذه التقنية ضمن صفحة تسجيل الخروج لضمان استعادة الذاكرة التي تحجزها حالة الجلسة بأسرع وقت.
Clear()	تقوم هذه الطريقة بإزالة جميع العناصر من غرض الجلسة لكن تقوم بالمحافظة على معرف الجلسة.

مثال على استخدام حالة الجلسة:

سنستخدم في هذا المثال حالة الجلسة لتخزين عدة أغراض مفروشات.

فيما يلي الرماز الخاص بتعريف الصف Furniture:

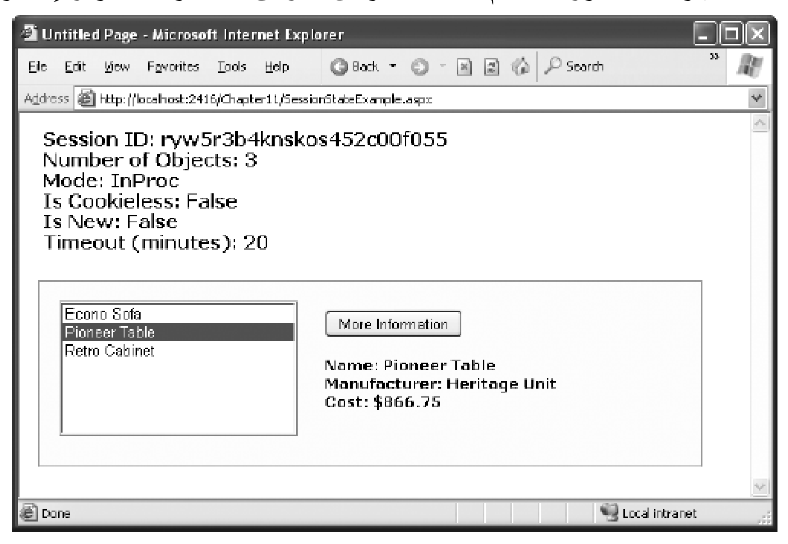
```
public class Furniture
{
public string Name;
public string Description;
```

```
public decimal Cost;
public Furniture(string name, string description,
decimal cost)
{
Name = name;
Description = description;
Cost = cost;
}
}
```

يتم إنشاء ثلاث أغراض مفروشات في المرة الأولى التي يتم فيها تحميل الصفحة وتخزن ضمن حالة الجلسة.

بإمكان المستخدم اختيار أحد المفروشات من قائمة.

عند اختيار أحد المفروشات تتم استعادة الغرض الموافق للعنصر المختار و إظهار معلوماته.



فيما يلي الرماز الخاص بالمثال:

```
public partial class SessionStateExample : System.Web.UI.Page
{
protected void Page_Load(Object sender, EventArgs e)
{
if (!this.IsPostBack)
{
// Create Furniture objects.
Furniture piece1 = new Furniture("Econo Sofa",
"Acme Inc.", 74.99M);
Furniture piece2 = new Furniture("Pioneer Table",
"Heritage Unit", 866.75M);
Furniture piece3 = new Furniture("Retro Cabinet",
"Sixties Ltd.", 300.11M);
// Add objects to session state.
Session["Furniture1"] = piece1;
Session["Furniture2"] = piece2;
Session["Furniture3"] = piece3;
// Add rows to list control.
lstItems.Items.Add(piece1.Name);
lstItems.Items.Add(piece2.Name);
lstItems.Items.Add(piece3.Name);
}
// Display some basic information about the session.
// This is useful for testing configuration settings.
lblSession.Text = "Session ID: " + Session.SessionID;
lblSession.Text += "<br />Number of Objects: ";
lblSession.Text += Session.Count.ToString();
lblSession.Text += "<br />Mode: " + Session.Mode.ToString();
lblSession.Text += "<br />Is Cookieless: ";
```

```
lblSession.Text += Session.IsCookieless.ToString();
lblSession.Text += "<br />Is New: ";
lblSession.Text += Session.IsNewSession.ToString();
lblSession.Text += "<br />Timeout (minutes): ";
lblSession.Text += Session.Timeout.ToString();
}
protected void cmdMoreInfo_Click(Object sender, EventArgs e)
{
if (lstItems.SelectedIndex == -1)
{
lblRecord.Text = "No item selected.";
}
else
{
// Construct the right key name based on the index.
string key = "Furniture" +
(lstItems.SelectedIndex + 1).ToString();
// Retrieve the Furniture object from session state.
Furniture piece = (Furniture)Session[key];

// Display the information for this object.
lblRecord.Text = "Name: " + piece.Name;
lblRecord.Text += "<br />Manufacturer: ";
lblRecord.Text += piece.Description;
lblRecord.Text += "<br />Cost: " + piece.Cost.ToString("c");
}
}
```

جعل حالة الجلسة أكثر قابلية للتوسع

يواجه مطورو الوب مشكلة أداء عند محاولة تخزين كمية كبيرة من المعلومات ضمن حالة الجلسة وبالأخص عند تزايد الاستخدام المتزامن للتطبيق من قبل عدد كبير من المستخدمين.

يمكن في هذه الحالة استخدام قاعدة البيانات لتخزين معلومات الجلسة.

تمكن هذه الطريقة من تخزين كمية كبيرة من البيانات و لفترات طويلة (أسابيع أو حتى أشهر).

تؤثر هذه الطريقة على الأداء بسبب الاضطرار إلى الوصول إلى قاعدة البيانات مع كل طلب لأي صفحة.

إعدادات حالة الجلسة:

يتم إعداد الجلسة عن عبر الملف web.config الخاص بالتطبيق (المتواجد ضمن المجلد الافتراضي للتطبيق).

يمكن هذا الملف من تحديد إعدادات متقدمة لحالة الجلسة.

يوضح النص التالي أهم الخيارات التي يمكن تحديدها للعنصر <sessionState>.

ملاحظة: لن تستخدم جميع الخيارات دفعة واحدة فهناك خيارات لا يمكن تطبيقها إلا من أجل خيار محدد للخاصة mode.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<system.web>
<!-- Other settings omitted. -->
<sessionState
cookieless="UseCookies" cookieName="ASP.NET_SessionId"
regenerateExpiredSessionId="false"
```

```
timeout="20"
mode="InProc"
stateConnectionString="tcpip=127.0.0.1:42424"
stateNetworkTimeout="10"
sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI"
sqlCommandTimeout="30"
allowCustomSqlDatabase="false"
customProvider=""
/>
</system.web>
</configuration>
```

الإعداد Cookieless:

يمكن اسناد أي من القيم المحددة ضمن الترقيم HttpCookieMode إلى هذا الخيار.

يوضح الجدول التالي قيم هذا الترقيم و وصف كل منها:

القيمة	الوصف
UseCookies	يتم استعمال الكعكات لتخزين محددات حالة الجلسة(حتى في حالة عدم دعم المستعرض للكعكات أو إلغاء تفعيلها من قبل مستخدم التطبيق). هذه القيمة هي القيمة الافتراضية.
UseUri	عند اسناد هذه القيمة إلى الإعداد mode سيتم تخزين محدد الجلسة ضمن URL ولن يتم استخدام الكعكات حتى في حال دعم المستعرض لها.
UseDeviceProfile	عند إسناد هذه القيمة للإعداد mode تقوم ASP.NET باختيار استخدام الكعكات أو عدم استخدامها بالاعتماد على معلومات الغرض BrowserCapabilities. نقطة الضعف في هذه الطريقة أنها لا تأخذ بعين الاعتبار اختيار مستخدم التطبيق تعطيل استخدام الكعكات على مستعرض يدعم استخدامها.
AutoDetect	تقوم ASP.NET في هذه الحالة بمحاولة تحديد كون المستعرض يدعم الكعكات بمحاولة تخزين واستعادة قيمة كعكة. تعتبر هذه الطريقة أفضل من سابقتها كونها تأخذ بعين الاعتبار تعطيل مستخدم التطبيق لاستخدام الكعكات على المستعرض.

مثال

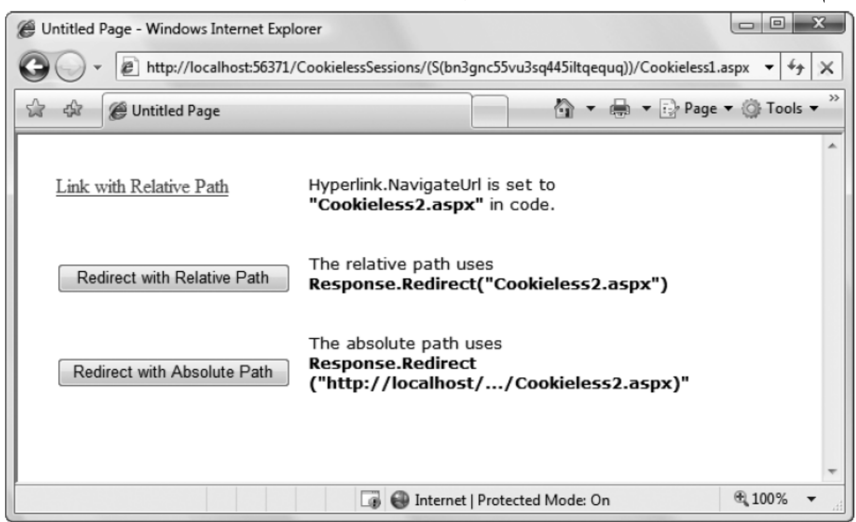
يوضح المثال التالي عمل حالة الجلسة التي لا تستخدم الكعكات.

يتضمن مثالنا صفحتي وب تستخدمان حالة الجلسة التي لا تعتمد الكعكات.

تحتوي الصفحة الأولى Cookieless1.aspx عنصر تحكم HyperLink وعنصري تحكم Buttons.

تقوم جميع عناصر التحكم السابقة بتوجيه المستعرض إلى الصفحة الثانية Cookieless2.aspx.

يقوم كل من هذه العناصر بتطبيق عملية إعادة التوجيه تلك بشكل مختلف.



يقوم عنصر تحكم HyperLink بإعادة توجيه المستعرض إلى العنوان المحدد ضمن الخاصة NavigateUrl.

يمثل هذا العنوان المسار النسبي إلى الصفحة Cookieless2.aspx.

عند الضغط على هذه الوصلة، نلاحظ أن الصفحة الهدف تمكنت من استعادة معلومات الجلسة مما يدل أن حالة الجلسة تعمل بشكل طبيعي في الوصلات التشعبية التي تستخدم المسار النسبي.

يستخدم الزران طريقة إعادة التوجيه البرمجية ()Response.Redirect للانتقال إلى الصفحة Cookieless2.aspx.

يكمن الاختلاف بين الزرين في استخدام الأول لمسار نسبي و الثاني لمسار مطلق.

نلاحظ أن الزر الذي يستخدم المسار النسبي يعمل بشكل سليم و يحافظ على معلومات الحالة دون أية خطوات إضافية.

```
protected void cmdLink_Click(Object sender, EventArgs e)
{
Response.Redirect("Cookieless2.aspx");
}
```

تضيع معلومات الحالة عند استخدام الزر الذي يعتمد المسار المطلق.

```
protected void cmdLinkAbsolute_Click(Object sender, EventArgs e)
{
string url = "http://localhost:56371/CookielessSessions/Cookieless2.aspx";
Response.Redirect(url);
```

الإعداد Timeout

يحدد هذا الخيار عدد الدقائق التي ستمر دون استقبال أي طلب من **مستخدم محدد** ليقوم ASP.NET بحذف حالة الجلسة.

تغيير هذا الإعداد قد يؤثر بشكل كبير على الأداء.

يكمن الخيار الأفضل في هذه المقايضة بإعطاء المستخدم الزمن الكافي للتوقف أثناء التفاعل مع الصفحات، لكن إبقاء هذا الزمن أصغرياً لتسريع عملية استعادة الذاكرة القيّمة التي تحجزها معلومات الحالة.

يمكن تغيير قيمة هذا الإعداد برمجياً عن طريق الخاصة Timeout كما يلي:

```
Session.Timeout = 10;
```

حالة التطبيق

تمكن حالة التطبيق من تخزين معلومات عامة يمكن الوصول إليها من قبل أي زبون.

تعتمد حالة التطبيق على الصف System.Web.HttpApplicationState والتي تتوفر ضمن جميع صفحات الوب عبر غرض Application المبيّت.

تشبه حالة التطبيق حالة الجلسة:

- نفس أنماط الأغراض
- تقوم بتخزين المعلومات على المخدم
- يستخدم نفس الصيغة المفهرسة للعناصر

مثال العداد العام

يعتبر مثال العداد العام أحد الأمثلة الشائعة لاستخدام حالة التطبيق.

يقوم هذا المثال بتتبع عدد مرات تنفيذ عملية معينة من قبل جميع الزبائن.

يمكن استخدام هذا العداد مثلاً لتتبع عدد مرات زيارة صفحة معينة.

```
protected void Page_Load(Object sender, EventArgs e)
{
// Retrieve the current counter value.
int count = 0;
if (Application["HitCounterForOrderPage"] != null)
{
count = (int)Application["HitCounterForOrderPage"];
}
// Increment the counter.
count++;
// Store the current counter value.
Application["HitCounterForOrderPage"] = count;
lblCounter.Text = count.ToString();
}
```

يتم تخزين معلومات حالة التطبيق كأغراض لذلك لابد من عملية تحويلها قسرياً عند استعادتها إلى النمط المطلوب.

لا يوجد زمن محدد لصلاحية معلومات حالة الجلسة إذ أنها تبقى مخزنة حتي يتم حذفها:

- باستخدام الرماز
- عند إعادة تشغيل التطبيق
- عند تحديث نطاق التطبيق آلياً بفعل تحديث رماز إحدى الصفحات أو مكوناتها

لا تستخدم حالة التطبيق بشكل كبير بسبب كونها غير فعالة.ففي مثالنا السابق مثلاً سنحصل على قيمة غير دقيقة للعداد عند الوصول بكثافة عالية إلى الصفحة.

يؤدي الوصول المتزامن إلى الصفحة إلى محاولة تعديل من أكثر من زبون على معلومات العداد في نفس الوقت والذي ينتهي إلى قيمة

خاطئة للعداد.

لمعالجة هذه المشكلة نستخدم الطريقة Lock() والطريقة Unlock.

يمنع استخدام Application.Lock() أي طلب غير الطلب الحالي من التعديل على معلومات الجلسة لحين انتهاء الطلب الحالي وتحرير غرض الجلسةApplication.Unlock.

```
protected void Page_Load(Object sender, EventArgs e)
{
// Acquire exclusive access.
Application.Lock();
int count = 0;
if (Application["HitCounterForOrderPage"] != null)
{
count = (int)Application["HitCounterForOrderPage"];
}
count++;
Application["HitCounterForOrderPage"] = count;
// Release exclusive access.
Application.Unlock();
lblCounter.Text = count.ToString();
}
```

بدائل استخدام حالة التطبيق

كما رأينا في المثال السابق،ستضطر جميع الطلبات للصفحة الحاوية على أي تعديلات على معلومات حالة التطبيق، إلى الانتظار لحين تحرير حالة التطبيق.

تؤدي هذه العملية إلى تراجع كبير في الأداء.

كان الهدف عندما تم اعتماد حالة التطبيق في النسخ السابقة لـــ ASP الوصول إلى وظيفتين أساسيتين:

1- تخزين ثوابت عامة يمكن الوصول إليها من قبل أي صفحة في التطبيق مثل ConnectionString.

2- تخزين المعلومات المستخدمة بكثرة و التي يتطلب إنشاؤها كلفة عالية (طاقة معالجة و زمن) لإعادة استخدامها عند اللزوم.

تم تطوير بدائل عن هاتين الوظيفتين باستخدام ملف web.config للوظيفة الأولى والخبء (caching)لتغطية الوظيفة الثانية.

مراجعة لخيارات إدارة الحالة

يبين الجدولان التاليان جميع خيارات إدارة الحالة و الفروق بينها.

	ViewState	Query String	Cookies
أنماط البيانات المسموح تخزينها	جميع أنماط بيانات Net. القابلة للسلسلة.	كمية محدودة من البيانات من نمط سلسة محرفية	سلسة محرفية
مكان التخزين	حقل خفي ضمن صفحة الوب	السلسلة المحرفية المخصصة لتخزين محدد URL ضمن المستعرض	ذاكرة الجهاز الزبون أو ملف نصي صغير بحسب إعدادات زمن حياة الكعكة.
زمن الحياة	مستمر بالنسبة لعملية إعادة إرسال لصفحة وحيدة.	تزول عند إدخال المستخدم عنوان URL جديد.	يتم تحديد الزمن من قبل المبرمج تستمر بين الجلسات ولعدة صفحات.

المجال	محدود بالصفحة الحالية	محدود بالصفحة الهدف	كامل التطبيق.
الأمان	غير قابلة للتعديل و لكن يمكن قراءتها بسهولة.يمكن تدعيمخا باستخدام الخاصة ViewStateEncryptionMode	واضحة بشكل مباشر للمستخدم يمكن قراءتها و تعديلها بسهولة.	غير آمنة يمكن تعديلها من قبل المستخدم.
التأثير على الأداء	تقوم بإبطاء التطبيق (زمن الإرسال و الاستقبال للصفحة) في حال تخزين كم كبير من المعلومات ، و لكنها لا تؤثر على أداء المخدم.	لا تؤثر لأن حجم البيانات صغير جداً	لا تؤثر لأن حجم البيانات صغير
الاستخدام المعتاد	الإعدادات المخصصة للصفحة	إرسال معرف منتج ما للوصول إلى صفحة التفصيلات.	تخصيص التفضيلات في موقع الوب.

	حالة الجلسة	**حالة التطبيق**
أنماط البيانات المسموح تخزينها	جميع أنماط بيانات NET.من أجل الوضع التخزين in-process. جميع أنماط بيانات NET. القابلة للسَلسلة في وضع التخزين out-of-process	جميع أنماط بيانات NET.
مكان التخزين	ذاكرة المخدم، خدمة الحالة أو مخدم SQL بحسب وضع التخزين المستخدم.	ذاكرة المخدم
زمن الحياة	الزمن المحدد ضمن Timeout القيمة التلقائية هي عشرون دقيقة.	عمر التطبيق ، عادة حت إعادة تشغيل التطبيق.
المجال	كامل التطبيق و لمستخدم محدد	كامل التطبيق و بعكس معظم الطرق فهي مرئية لجميع المستخدمين.
الأمان	آمنة جداً لأن المعلومات لا يتم إرسالها إلى الزبون.	آمنة جداً لأن المعلومات لا يتم إرسالها إلى الزبون.
التأثير على الأداء	تسبب بطء عند تخزين كمية كبيرة من المعلومات و بالأخص عند وجود عدد كبير من المستخدمين.	تسبب بطء عند تخزين كمية كبيرة من المعلومات لأن المعلومات ليس لها عمر محدد و ستستمر بالحياة ما حيا التطبيق.
الاستخدام المعتاد	تخزين معلومات سلة التسوق	تخزين أي معلومات عامة.

الفصل العاشر

عناصر تحكم التحقق من الصحة

يتعرف الدارس في هذا الفصل على:

- الغرض من استعمال عناصر تحكم التحقق من الصحة
- أنواع و استخدام عناصر التحقق من الصحة
- التحقق من طرف المخدم و التحقق من طرف الزبون
- صيغة التعبيرات النظامية واستخدامها ضمن عناصر تحكم التحقق

التحقق من الصحة

يمكن للمستخدم أن يرتكب أخطاءاً أثناء التعامل مع التطبيق كأن يقوم بـــ:

- إهمال إدخال قيمة في أحد الحقول ضمن النموذج.
- إدخال بريد الكتروني غير صالح.
- إدخال قيمة غير عددية في حقل مخصص لقيمة عددية.
- قد يقوم مستخدم باستغلال نقطة ضعف ضمن الرماز الخاص بالتطبيق ويقوم بإدخال معلومات مغلوطة بهيكلية محددة بحيث تؤدي إلى إعادة معلومات حساسة.

أحد الأمثلة على هذه المقاربة هجوم حقن SQL الذي يزود فيه المستخدم قيماً تقود إلى تشكيل أمر إلى قاعدة البيانات بشكل ديناميكي.

عناصر تحكم التحقق من الإدخال

توفر ASP.NET خمس عناصر تحقق من الإدخال أربعة منها مخصصة لنوع معين من التحقق في حين يوفر العنصر الخامس إمكانية كتابة إجرائيات تحقق مخصصة.

توفر ASP.NET أيضاً عنصر تحكم خاص ValidationSummary بإظهار رسائل الخطأ في الإدخال.

يوضح الجدول التالي عناصر التحكم تلك مع وصف كل منها:

صف عنصر التحكم	الوصف
RequiredFieldValidator	تنجح عملية التحقق في عنصر التحكم هذا في حال احتواء دخل عنصر التحكم على أي سلسلة محرفية.
RangeValidator	تنجح عملية التحقق في عنصر التحكم هذا في حال احتواء دخل عنصر التحكم على قيمة ضمن نطاق رقمي أو حرفي أو زمني

	معين.
CompareValidator	تنجح عملية التحقق في عنصر التحكم هذا في حال احتواء دخل عنصر التحكم على قيمة موافقة لقيمة في عنصر تحكم آخر أو قيمة ثابتة يتم تحديدها.
RegularExpressionValidator	تنجح عملية التحقق في عنصر التحكم هذا في حال تطابق دخل عنصر التحكم مع تعبير نظامي محدد.
CustomValidator	يتم في عنصر التحكم هذا إجراء عملية التحقق بالاعتماد على إجرائية مخصصة من قبل المستخدم.

ملاحظة: يمكن ربط كل عنصر تحقق بعنصر إدخال وحيد كما يمكن ربط أكثر من عنصر تحقق بعنصر إدخال واحد لتوفير إمكانية تحقق من عدة أنماط.

ستنجح عملية التحقق في حال استخدام عناصر تحكم التحقق RangeValidator ، CompareValidator أو RegularExpressionValidator مع عنصر تحكم إدخال لايحتوي أي قيمة. لذلك يفضل عند استخدام عناصر تحكم التحقق تلك إضافة عنصر تحكم تحقق RequiredFieldValidator.

التحقق من جهة المخدم

يمكن استخدام عناصر تحكم التحقق بشكل آلي للتحقق من صفحة عند إرسالها أو بشكل يدوي ضمن النص البرمجي.
تعتبر المقاربة الأولى الأكثر شيوعاً.

عند استخدام التحقق بشكل آلي يقوم المستخدم بإدخال المعلومات ضمن عناصر تحكم الإدخال وحال انتهائه يقوم بإرسال المعلومات بالنقر على زر لإرسال الصفحة. يمتلك كل زر الخاصة CausesValidation والتي تأخذ القيمة true أو false.

يعتمد السيناريو الناتج عن نقر المستخدم على زر الإرسال على قيمة الخاصة CausesValidation:

- في حال كانت قيمة هذه الخاصة false تقوم ASP.NET بإهمال عناصر التحقق من الإدخال المرتبطة بعنصر الإدخال، يتم إرسال الصفحة وتنفيذ الرماز الخاص بمقبض حدث الإرسال بشكل طبيعي.
- أما في حال كانت قيمة الخاصة CausesValidation هي true تقوم ASP.NET آلياً بتقييم الصفحة عند النقر على زر الإرسال حيث تقوم بالتحقق من كل عنصر على الصفحة. إذا فشل التحقق من أي عنصر تحكم ستعاد الصفحة مع معلومات الخطأ وبحسب الإعدادات الخاصة بإظهار أخطاء الإدخال في كل عنصر تحكم. في هذه الحالة قد لا يتم تشغيل الرماز الخاص بمقبض حدث الإرسال، لذلك لابد من التأكد ضمن مقبض الحدث من نجاح عملية التحقق أو فشلها.

بناءً على هذا الوصف ندرك أن عملية التقييم تتم بشكل آلي عند ضغط الأزرار وليس عند إعادة إرسال الصفحة من خلال حدث change أو عند ضغط الأزرار التي تكون فيها قيمة الخاصة CausesValidation مساوية لـ false.

على كل حال يمكننا التحقق من صحة عنصر تحكم يدوياً (باستخدام النص البرمجي) ومن ثم اتخاذ القرار بناءً على نتيجة التقييم.

التحقق من جهة الزبون

تقوم ASP.NET آلياً بإضافة رماز JavaScript من طرف الزبون لإتمام عملية التحقق في حال تم استخدام المستعرضات الحديثة مثل Firefox أو Internet Explorer.

عند ضغط المستخدم على زر (تم إسناد قيمة true إلى الخاصة CausesValidation فيه) سيتم إظهار رسالة التحقق على الصفحة دون الحاجة إلى إرسال الصفحة إلى المخدم. تؤدي هذه العملية إلى استجابة صفحة الوب.
لن يمنع تجاوز التحقق من الصحة بنجاح من جهة الزبون ASP.NET من إعادة إرسال الصفحة إلى المخدم. تؤمن هذه المقاربة حماية من عمليات التجاوز المقصودة للتحقق من الصحة التي قد يقوم بها مستخدم بهدف اختراق التطبيق.
إن عملية تجاوز التحقق من الصحة من طرف الزبون هي عملية سهلة إذ يمكن للزبون إعادة صياغة أو كتابة أو الاطلاع على منطق الرماز أو الحقول المستخدمة لإرسال البيانات في حين تكون هذه العملية صعبة جداً عند استخدام الرماز من جهة المخدم.
تؤمن عملية الدمج بين التحقق من الصحة من جهة الزبون ومن جهة المخدم مما يوفر أعلى استجابة ممكنة دون التضحية بأمان التطبيق.

عناصر تحكم التحقق

تعتمد عناصر تحكم التحقق على فضاء الأسماء System.Web.UI.WebControls وتقوم بالوراثة من الصف BaseValidator.
يعرّف الصف BaseValidator مجموعة من الخصائص والوظائف نوضحها في الجدول التالي:

الخاصة	الوصف
ControlToValidate	تحدد هذه الخاصة عنصر التحكم الذي سيقوم عنصرالتحقق بتقييمه. يمكن لكل عنصر تحقق تقييم عنصر إدخال وحيد كما يمكن استخدام أكثر من عنصر تحقق لتقييم عنصر إدخال وحيد.
ErrorMessage and ForeColor	تمثل هذه الخاصة الرسالة التي سيتم إظهارها في حال فشل عملية التحقق من الصحة. تمكن الخاصة ForeColor من إظهار الرسالة بلون مميز.
Display	تمكن هذه الخاصة من تحديد مكان إظهار رسالة التحقق: – يمكن أن يتم إدراجها بشكل ديناميكي ضمن الصفحة عند إسناد القيمة Dynamic إلى الخاصة Display. تفيد هذه الوضعية في حال الرغبة باستخدام أكثر من عنصر تحقق لنفس عنصر الإدخال حيث تتراكم رسائل الخطأ بشكل ديناميكي. – في حال إسناد القيمة Static إلى الخاصة Display سيتم حجز حيز خاص للرسالة.تستخدم هذه الطريقة عند وجود تصميم ثابت لايرغب المطوّر بتغييره نتيجة الظهور الديناميكي لرسالة الخطأ.
IsValid	تعيد هذه الخاصة القيمة true أو false اعتماداً على نجاح التحقق أو فشله. عادة يتم التحقق من الخاصة IsValid لكامل الصفحة عوضاً عن التحقق من هذه القيمة لكل عنصر تخقق فيها.
Enabled	يتم تعطيل التحقق من الصحة عند إرسال الصفحة لعنصر

	التحكم في حال إسناد القيمة false إلى هذه الخاصة.
EnableClientScript	عند إسناد القيمة true إلى هذه الخاصة تقوم ASP.NET بإضافة نصوص JavaScript و DHTML لتمكين التحقق من الصحة من جهة الزبون على المستعرضات التي تدعم هذه التقنيات.

الخصائص الفريدة لعناصر تحكم التحقق

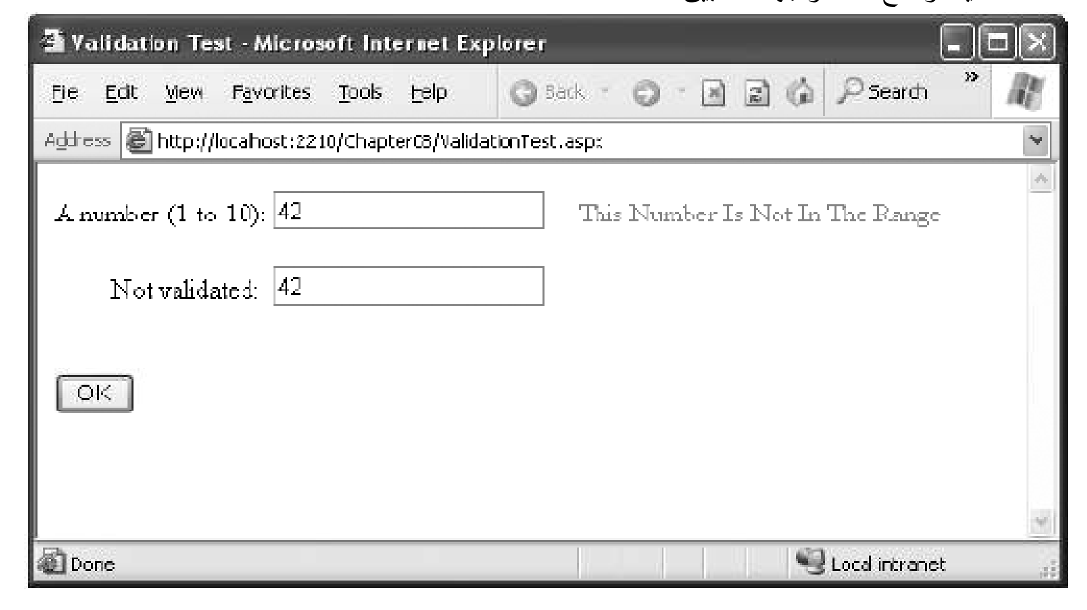
عند استخدام عناصر تحكم التحقق لا بد من إعطاء قيمة للخاصة ControlToValidate والخاصة ErrorMessage كما تتفرد عناصر تحكم الحقق بخصائص فريدة تختلف بها عن بقية العناصر نوضحها في الجدول التالي:

عنصر تحكم التحقق	الخصائص الفريدة الإضافية
RequiredFieldValidator	–
RangeValidator	MaximumValue, MinimumValue, Type
CompareValidator	ControlToCompare, Operator, Type, ValueToCompare
RegularExpressionValidator	ValidationExpression
CustomValidator	ClientValidationFunction, ValidateEmptyText, ServerValidate event

مثال بسيط على عناصر تحكم التحقق

يستخدم المثال التالي عنصر تحقق RangeValidator للتحقق من إدخال قيمة ضمن المجال المسموح ضمن عنصر تحكم النص.

اللقطة التالية توضح شكل واجهة التطبيق:



يتضمن هذا التطبيق عنصري تحكم علبة نصية إضافة إلى عنصر تحكم التحقق و زر الإرسال.

فيما يلي النص البرمجي للصفحة:

```
<asp:TextBox id="txtValidated" runat="server" />
<asp:RangeValidator id="RangeValidator" runat="server"
ErrorMessage="This Number Is Not In The Range"
ControlToValidate="txtValidated"
MaximumValue="10" MinimumValue="1"
Type="Integer" />
<br /><br />
Not validated:
<asp:TextBox id="txtNotValidated" runat="server" /><br /><br />
<asp:Button id="cmdOK" runat="server" Text="OK" OnClick="cmdOK_Click" />
<br /><br />
<asp:Label id="lblMessage" runat="server" EnableViewState="False" />
```

أما النص البرمجي الخاص بمقبض حدث ضغط الزر:

```
protected void cmdOK_Click(Object sender, EventArgs e)
{
lblMessage.Text = "cmdOK_Click event handler executed.";
}
```

عند اختبار هذا المثال باستخدام مستعرض حديث سنلاحظ أن استجابة الصفحة بإظهار رسالة الخطأ بعد إدخال قيمة خارج النطاق بعد ضغط الزر Tab دون إعادة إرسال للصفحة.

تم الحصول على هذه الميزة من خلال إضافة Asp.NET لنص لغة JavaScript من جهة المخدم الذي قام بالاستجابة للحدث الخاص بتغيير التركيز من جهة الزبون.

إن إضافة هذا النص البرمجي من جهة الزبون لا تعد عملية بسيطة و تستهلك بعض الوقت.

تتم العناية بكل هذه التفاصيل من قبل ASP.NET بشكل آلي.

ذكرنا أن النص البرمجي من جهة الزبون غير مدعوم من جميع المستعرضات.

لاختبار ماذا سيحصل في حال استخدام مستعرض لايدعم JavaScript يكفي إسناد القيمة false إلى الخاصة RangeValidator.EnableClientScript.

في هذه الحالة لن تحدث أي عملية اختبار للصحة قبل ضغط زر الإرسال وإرسال الصفحة إلى المخدم حيث تتم إعادة الصفحة مع رسالة الخطأ الناتجة عن عملية التحقق على المخدم.

يرتبط السيناريو السابق بخلل لا بد من تجاوزه و هو أن النص البرمجي الخاص بمقبض الحدث سيتم تنفيذه سواء كانت نتيجة اختبار الصحة سلبية أم إيجابية.

لذلك يجب قسر إيقاف تنفيذ رماز مقبض الحدث في حال عدم تجاوز اختبار تحقق الصحة بنجاح.

تتم هذه العملية باستخدام النص البرمجي وبالاعتماد على الخاصة IsValid كما يظهر في الرماز التالي:

```
protected void cmdOK_Click(Object sender, EventArgs e)
{
// Abort the event if the control isn't valid.
if (!RangeValidator.IsValid) return;
lblMessage.Text = "cmdOK_Click event handler executed.";
}
```

كما يمكن الاعتماد على قيمة الخاصة IsValid لغرض الصفحة لضمان تجاوز اختبار الصحة لجميع عناصر تحكم التحقق على هذه الصفحة.

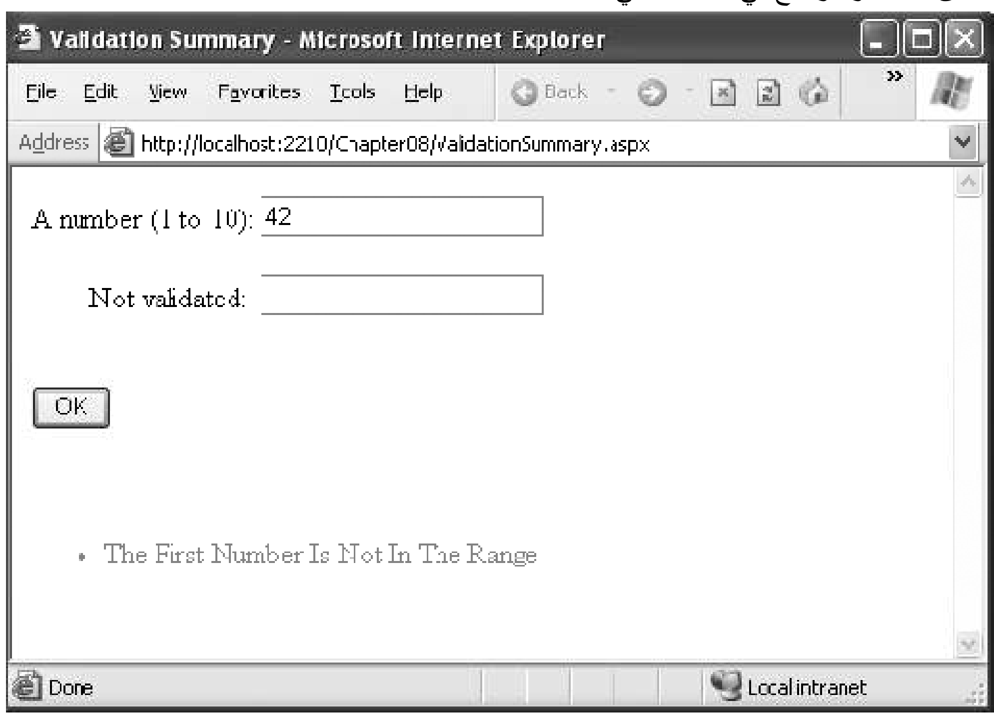
```
protected void cmdOK_Click(Object sender, EventArgs e)
{
// Abort the event if any control on the page is invalid.
if (!Page.IsValid) return;
lblMessage.Text = "cmdOK_Click event handler executed.";
}
```

خيارات الإظهار الأخرى:

يمكن في بعض الحالات أن يكون المطور قد قام بإنشاء تصميم لصفحة تتضمن أكثر من حقل.
قد يؤدي إظهار نتيجة اختبار الصحة بشكل ديناميكي في مثل هذه الحالة إلى إعادة تنسيق غير مرغوب فيها.
يوفر عنصر التحكم ValidationSummary حل مناسب لهذا السيناريو.
لاختبار عنصر التحكم هذا يجب إسناد القيمة None إلى الخاصة Display ضمن عنصر تحكم التحقق RangeValidator (في مثالنا). سيضمن هذا عدم ظهور رسالة الخطأ.
لايعني هذا أن عملية التحقق من الصحة لن تتم وسيتم منع المستخدم من إدخال قيم خاطئة.
لإظهار نتائج التحقق من الصحة يجب إضافة عنصر التحكم ValidationSummary إلى موقع مناسب في الصفحة بحيث لايؤثر على التصميم:

```
<asp:ValidationSummary id="Errors" runat="server" />
```

عند النقر على زر OK (في مثالنا) سيتم إظهار لائحة تتضمن جميع رسائل الخطأ التي تم استقبالها من جميع عناصر تحكم التحقق. كما هو موضح في الشكل التالي:



خيارات الإظهار الأخرى (تتمة):

يمكن استخدام الخاصة Text لعنصر تحكم التحقق لإظهار إشارة تدل على الخطأ في حقل الإدخال حتى في حال استخدام عنصر التحكم ValidationSummary.

عادة ما يتم استخدام إشارة (*) كما يوضح الرماز التالي:

```
<asp:RangeValidator id="RangeValidator" runat="server"
Text="*" ErrorMessage="The First Number Is Not In The Range"
ControlToValidate="txtValidated"
MaximumValue="10" MinimumValue="1" Type="Integer" />
```

أو بإظهار أيقونة تعبر عن حصول خطأ كما في الرماز:

```
<asp:RangeValidator id="RangeValidator" runat="server"
Text="<img src='ErrorIcon.gif' />" alt='Error' ... />
```

ملاحظة: لا بد من الإشارة هنا إلى ضرورة إسناد قيمة مغايرة لـــ None إلى الخاصة Display لعناصر تحكم التحقق عند الرغبة بإظهار إشارات الخطأ المذكور أعلاه.

يملك عنصر التحكم ValidationSummary خصائص أخرى مفيدة مثل الخاصة HeaderText التي تظهر عنوان خاص أعلى لائحة رسائل الخطأ. كما يمكن تغيير لون الإظهار باستخدام الخاصة ForeColor واختيار نمط الإظهار باستخدام الخاصة DisplayMode.

تحدد الخاصة DisplayMode إظهار اللائحة على شكل لائحة أو لائحة نقطية أو على شكل مقطع وحيد وذلك بإسناد القيم التالية List و BulletList و SingleParaghraph.

أخيراً يمكن استخدام الخاصة ShowMessageBox لإظهار رسائل التحقق من الصحة ضمن نافذة منبثقة.

يستخدم هذا الخيار عند الرغبة بعدم المس نهائياً بالتصميم ضمن الصفحة الحاوية على عناصر التحقق.

يجبر هذا الخيار المستخدم على النقر على زر OK ضمن نافذة الرسالة لاستمرار التفاعل مع الصفحة، مما يجعل هذا الخيار غير مناسب في حال أراد المستخدم المحافظة على إظهار نتيجة التحقق.

التحقق من الصحة بشكل يدوي

أحد الخيارات المتاحة في عملية التحقق هي إجراء العملية بشكل يدوي و لكن باستخدام عناصر تحكم التحقق.

تمكن هذه الطريقة من الحصول على خيارات أوسع و أخذ قيم أخرى بعين الاعتبار عند إجراء عملية التحقق من الصحة.

لا بد عند اختيار العمل باستخدام هذه الطريقة من إسناد القيمة false إلى الخاصة EnableClientScript لعناصر تحكم التحقق.

كما لا بد من إسناد القيمة false إلى الخاصة CausesValidation والاستعاضة عنها باستخدام الطريقة Page.Validate() لتقييم الصفحة بشكل يدوي.

المثال التالي يوضح هذه المقاربة حيث يتم فحص جميع عناصر التحقق ضمن الصفحة و اختبار قيمة IsValid لكل منها:

```
protected void cmdOK_Click(Object sender, EventArgs e)
{
string errorMessage = "<b>Mistakes found:</b><br />";
// Search through the validation controls.
foreach (BaseValidator ctrl in this.Validators)
{
if (!ctrl.IsValid)
```

```
{
errorMessage += ctrl.ErrorMessage + "<br />";
// Find the corresponding input control, and change the
// generic Control variable into a TextBox variable.
// This allows access to the Text property.
TextBox ctrlInput =
(TextBox)this.FindControl(ctrl.ControlToValidate);
errorMessage += " * Problem is with this input: ";
errorMessage += ctrlInput.Text + "<br />";
}
}
lblMessage.Text = errorMessage;
}
```

اختبار الصحة اعتماداً على التعابير النظامية

يعد استخدام التعابير النظامية في اختبار الصحة من أقوى الطرق التي توفرها ASP.NET.

تمكن هذه الطريقة من مطابقة سلسلة محرفية ما مع نموذج نمطي معين مثل بريد الكتروني،رقم هاتف، اسم ملف.

تستخدم التعابير النظامية بشكل كبير في معظم لغات التطوير البرمجية حتى أنها يمكن أن تعتبر لغة بحد ذاتها و تغطي كتاب بكامله إلا أننا سنحاول تغطية أساسيات استخدامها.

تتألف جميع التعابير النظامية من نوعين أساسيين من المحارف و المحارف الوصفية. لا تختلف المحارف عن سلاسل المحارف التي تستخدمها في نصوص اللغات البرمجية، إذ تعبر عن محارف محددة مثل حرف "l" مثلاً و لا شيء غير هذا المحرف.

في حين يكمن سر التعبيرات النظامية في المحارف الوصفية حيث يعبر محرف وصفي واحد مثل "*" مثلاً عن جميع المحارف الرقمية والأبجدية.

يوضح الجدول التالي المحارف الوصفية الأساسية و عمل كل منها:

المحرف	الوصف
*	عدم ورود أو أكثر من ورود للمحرف أو التعبير الجزئي السابق. مثلاً 7*8 تطابق 7778 أو 8
+	ورود أو أكثر للمحرف أو التعبير الجزئي السابق، مثلاً: 7+8 تطابق 7778 و لا تطابق 8
( )	تجمع تعبير جزئي ليتم معاملته كعنصر وحيد ضمن التعبير النظامي، مثلاً: +(78) تطابق 78 و تطابق 787878
{m,n}	تحدد ورود المحرف أو التعبير الجزئي السابق من m إلى n مرة، مثلاً: A{1,3} تطابق A و AA و AAA
|	تطابق مع أي من طرفيها مثلاً: 8|6 تطابق 8 أو 6
[ ]	تطابق محرف أو أكثر من مجال المحارف المحدد، مثلاً: [A-C] تطابق A أو B أو C
[^]	تطابق محرف ليس ضمن المجال المحدد، مثلاً:

	[^A-B] تطابق أي محرف عدا A و B
.	أي محرف عدا محرف السطر الجديد، مثلاً: تطابق here. الكلمة where و there
\s	أي محرف فارغ مثل محرف الجدولة Tab أو محرف الفراغ Space.
\S	تطابق أي محرف ماعدا محارف الفراغ.
\d	أي محرف رقمي.
\D	أي محرف غير رقمي.
\w	أي كلمة محرفية (حرف، رقم ،شرطة سفلية).
\W	أي محرف عدا كلمة محرفية.

بعض التعابير النظامية الشائعة

يبين الجدول التالي مجموعة من التعابير النظامية الشائعة الاستخدام:

المحتوى	التعبير النظامي	الوصف
عنوان بريد الكتروني	\S+@\S+\.\S+	يتم التأكد من وجود إشارة @ و إشارة النقطة دون السماح بوجود فراغات
كلمة سر	\w+	أي تسلسل كلمات محرفية (أحرف،فراغات و شرطة سفلية)
كلمة سر بطول محدد	\w{4,10}	طول كلمة السر بين 4 إلى 10 محارف
كلمة سر متقدمة	[a-zA-Z]\w{3,9}	تم تحديد طول لكلمة السر بين 4 و 10 محارف على أن يكون المحرف الأول حرف عادي من a-z أو من A-Z
كلمة سر متقدمة أخرى	[a-zA-Z]\w*\d+\w*	في هذه الكلمة هناك شرط للبدء بمحرف متبوعاً بكلمات حرفية ثم رقم أو أكثر ثم صفر أو أكثر من محرف.
حقل بطول محدد	\S{4,10}	يسمح بإدخال 4 إلى 10 محارف و لكن يسمح بالمحارف الخاصة
الرقم التأميني (بحسب التنسيق المستخدم في الولايات المتحدة)	\d{3}-\d{2}-\d{4}	تسلسل من ثلاث أرقام ثم رقمين ثم أربعة أرقام مفصولة بشرطة.

مجموعات التحقق

قد تحتوي الصفحات المعقدة على أكثر من مجموعة منفصلة من عناصر التحكم الموزعة أحياناً على أكثر من عنصر تحكم Panel.

في مثل هذا السيناريو قد يرغب المطور في إجراء عملية تحقق من الصحة لكل مجموعة بشكل منفصل.

مثلاً يمكن أن نقوم بإنشاء نموذج يتضمن عناصر تحكم تسجيل الدخول كما يتضمن في حيز منفصل عناصر التحكم الخاصة بتسجيل مستخدم جديد.

يحتوي كل جزء من جزئي النموذج المذكورين زر إرسال خاص.يتم إجراء التحقق من الصحة بناء على نقر أي من زري الإرسال.

لكن عملية التحقق يجب أن تتم فقط لتلك العناصر الخاصة بذلك الجزء من النموذج.

يمكن تحقيق هذا السيناريو باستخدام مجموعات التحقق.

لإنشاء مجموعات التحقق يكفي وضع عناصر تحكم الإدخال،عناصر التحكم و زر الإرسال (الذي تم تفعيل خاصة CausesValidation فيه) ضمن نفس المجموعة المنطقية.

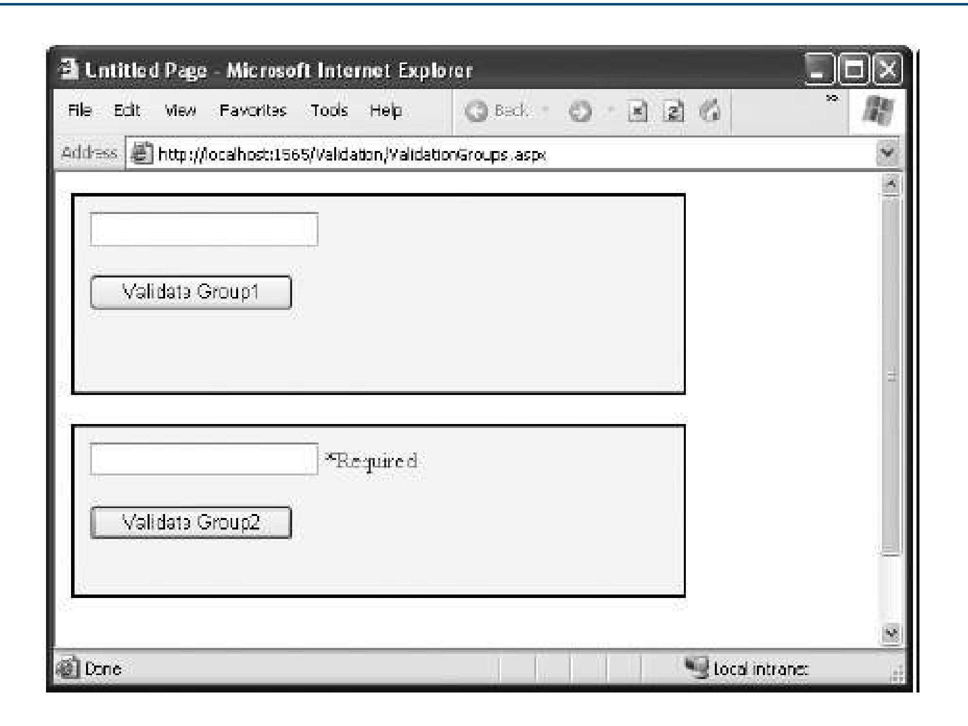
يتم الوصول إلى هذا الغرض بإسناد اسم المجموعة المنطقية و ليكن مثلاً loginGroup إلى الخاصة ValidationGroup لجميع عناصر التحكم التابعة لهذه المجموعة المنطقية بما يتضمن زر الإرسال.

يوضح المثال التالي هذه الفكرة:

```
<form id="form1" runat="server">
<asp:Panel ID="Panel1" runat="server">
<asp:TextBox ID="TextBox1" ValidationGroup="Group1" runat="server" />
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
ErrorMessage="*Required" ValidationGroup="Group1"
runat="server" ControlToValidate="TextBox1" />
<asp:Button ID="Button1" Text="Validate Group1"
ValidationGroup="Group1" runat="server" />
</asp:Panel> <br />
<asp:Panel ID="Panel2" runat="server">
<asp:TextBox ID="TextBox2" ValidationGroup="Group2"
runat="server" />
<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
ErrorMessage="*Required" ValidationGroup="Group2"
ControlToValidate="TextBox2" runat="server" />
<asp:Button ID="Button2" Text="Validate Group2"
ValidationGroup="Group2" runat="server" />
</asp:Panel>
</form>
```

عند نقر زر validate Group1 سيتم التحقق من صحة العلبة النصية الأولى فقط وكذلك الأمر بالنسبة للعلبة النصية ضمن اللوحة Panel2Panel2.

ملاحظة: في حال استخدام عنصر تحكم ضمن هذا السيناريو دون تحديد المجموعة المنطقية التي ينتمي إليها سيتم التحقق من صحته عند التحقق من صحة أي مجموعة منطقية أخرى.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost:1565/Validation/ValidationGroups.aspx
Validate Group1
*Required
Validate Group2
Done
Local intranet

الفصل الحادي عشر

إدارة البيانات في NET

إدارة البيانات في ASP.NET

سنغطي في هذه الجلسة المواضيع المتعلقة بإدارة البيانات في ASP.NET. ونقصد هنا بتعبير إدارة البيانات عمليات الوصول إلى البيانات المخزنة في ملفات وتطبيقات أخرى ومعالجتها.

نسمي في الإطار العام، مصادر المعلومات بمخازن البيانات.

يتضمن إطار عمل NET. مجموعة من الصفوف التي تتبنى تقنيات وصول متقدمة إلى البيانات المصممة خصيصاً للاستخدام مع (NET.).

مخازن البيانات والوصول إليها

يرتبط مفهوم إدارة البيانات بمصادر البيانات العلائقية مثل قواعد البيانات، ولكن تقنيات إدارة البيانات في NET. تقدم إمكانيات أخرى متميزة منها: الاتصال والتعامل مع ملفات XML والتقنيات المرتبطة بها.

تاريخياً، كانت قواعد البيانات عادة مبنية على ملف وتستخدم طول سجل محدد كما هي الحال في ملفات txt.

إذ كانت تجري قراءة الملفات إلى جداول من قبل برامج قواعد البيانات أو تقنيات الوصول إلى البيانات، وكانت تُطبق قواعد موجودة في ملفات أخرى لربط سجلات من جداول مختلفة بعضها ببعض. بعد نضوج هذه التقنيات ظهرت قواعد البيانات العلائقية لتوفر طرق تخزين أفضل مع الطول الديناميكي للسجل و تقنيات وصول أكثر فعالية للبيانات. على أي حال بقي مكان التخزين الأساسي هو قاعدة البيانات.

الانتقال إلى البيئة الموزعة

في السنوات الأخيرة، تغيرت العديد من المتطلبات وآليات عمل التطبيقات في معظم الأعمال وتم الابتعاد عن مفهوم قاعدة البيانات العلائقية المركزية، وأصبحت البيانات موزعة بين مخدمات البريد الالكتروني ووثائق المكتب وأماكن ووسائط أخرى، ضمن قواعد البيانات أيضاً.

إدارة البيانات في NET.

توجد تقنيات جديدة للوصول إلى البيانات، تتناسب مع البيئة الجديدة الموزعة التي تكلمنا عنها. لذا سنستعرض ما تقدمه NET. فعلياً في هذا المجال.

سنبدأ بإعطاء لمحة عامة عن جميع صفوف إدارة البيانات في NET. لنرى كيف تنسجم جميع أغراض إدارة البيانات مع بيئة البرمجة المهيكلة التي تقدمها (NET.).

فضاء الأسماء:
تُبنى جميع صفوف إدارة البيانات العلائقية على فضاء الأسماء system.data ويطلق عادة ado.net على فضاءات الأسماء الموجودة في الجدول التالي:

فضاء الأسماء	الوصف
System.data	يتضمن جميع الأغراض الأساسية المستخدمة للوصول إلى، وتخزين البيانات في قواعد البيانات العلائقية. من هذه الأغراض: Dataset و DataTable و DataRelation. تكون كل من هذه الأغراض مستقلة عن نمط مصدر البيانات والطريقة التي نتصل بها بهذا المصدر.
System.data.common	تحتوي الصفوف الأساسية المستخدمة من الأغراض الأخرى وخاصة الأغراض العامة من فضاء الأسماء OleDb و SqlClient. بصورة عامة لا نقوم باستيراد فضاء الأسماء هذا في تطبيقاتنا.
System.data.OleDb	يحتوي الأغراض التي تُستخدم للاتصال مع مصدر البيانات باستخدام مزود Ole-Db مثل OleDbConnection ، OleDbCommand. ترث هذه الأغراض طرق وخصائص من الصفوف المشتركة
System.data.SqlClient	تحتوي الأغراض التي يمكننا استخدامها للاتصال مع مصادر البيانات عبر سياق من البيانات الجدولية الخاصة ب SQL Server فقط. حيث توفر أداء أفضل بإزالتها بعض الطبقات الوسيطة المطلوبة من اتصال OLE_DB. ترث الأغراض مثل SQLConnection و SQLCommand من الصفوف المشتركة كـــ OleDb الخصائص و الطرق و الأحداث.
System.Data.SqlType	تحتوي الصفوف اللازمة لاستخدام أنماط البيانات في قواعد البيانات العلائقية مثل SQLServer و المختلفة عن تلك القياسية في NET. كأغراض SqlDateTime. SqlMoney و SqlBinary. يحسن استخدام هذه الأغراض الأداء بشكل ملحوظ ويقلل أخطاء التحويل بين أنماط البيانات.

هناك أيضاً سلسلة من فضاءات الأسماء الحاوية على صفوف يمكن استخدامها للتعامل مع ملفات XML بدلاً من استخدام قواعد البيانات العلائقية، تكون هذه الأسماء مبنية على System.Xml.

فضاء الأسماء	الوصف
System.Xml	تحتوي الأغراض الأساسية اللازمة لإنشاء، وقراءة ، وتخزين، وكتابة ومعالجة ثائق XML بحسب توصيات W3C. تحتوي XmlDocument بالإضافة إلى سلسلة من الأغراض التي تمثل أنواع مختلفة من العقد في وثيقة XML.
System.Xml.Schema	تحتوي الأغراض المسؤولة عن إنشاء، وتخزين، ومعالجة، الهيكل والعقد المحتواة في هيكل وثيقة XML.
System.Xml.Sirialization	يحتوي مجموعة من الأغراض التي يمكن استخدامها لتحويل وثيقة XML إلى تنسيقات أخرى مختلفة مثل SOAP للنقل عبر الشبكة مثلاً.
System.Xml.Xpath	يحتوي الصفوف اللازمة لتطبيق عمليات القراءة، والتخزين والكتابة واستعلام عن وثائق XML باستخدام غرض مبني على XPath. تتضمن أغراض مثل XpathDocument و XpathNavigator والأغراض التي تمثل تعبيرات XPath.
System.Xml.Xsl	يحتوي الأغراض اللازمة لعملية تحويل ملف XML إلى تنسيقات أخرى باستخدام XSL و XSLT. يكون الغرض الأساسي فيه هو XslTransform

استيراد فضاءات الأسماء اللازمة للعمل مع مصادر البيانات

- لا بد للصفحات التي تستخدم أغراضاً من مكتبة صفوف إطار العمل NET. أن تستورد فضاءات الأسماء الحاوية على الأغراض التي تريد إنشاء مثيل عنها. يجري استيراد الكثير من هذه الفضاءات تلقائياً
- لكن فضاء أسماء إدارة البيانات ليس من تلك الفضاءات التي يتم استيرادها تلقائياً، لذا يجب علينا استيراده بصورة صريحة في النص البرمجي

للوصول إلى قواعد البيانات العلائقية لا بد لنا من استخدام فضاء الأسماء System.Data على الأقل و أيٍ من System.Data.OleDb أو System.Data.SqlClient اعتماداً على الطريقة التي نود الاتصال بها مع مصدر البيانات و ذلك بالصيغة:

```
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb" %>
```

أو

```
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient" %>
```

يمكن في VB.NET استخدام Imports و في C# استخدام Using

هناك حالات خاصة نضطر فيها لاستيراد فضاءات أسماء أخرى كحالة إنشاء عنصر من الغرض DataTableMapping، حيث يتوجب علينا استيراد فضاء الأسماء System.Data.Common. كما يتوجب علينا استيراد System.Data.SqlType عندما نريد استخدام أنماط بيانات من الأنواع التي يستخدمه SqlServer

للوصول إلى بيانات XML باستخدام الأغراض في صفوف مكتبات Net.، يمكننا غالباً غض النظر عن استيراد فضاء الأسماء الأساسي System.Xml .

على كل حال سنضطر إلى استيراد فضاء الأسماء System.Xml.Xpath عندما نريد إنشاء غرض XpathDocument.

وسنضطر أيضاً لاستيراد فضاء الأسماء System.Xml.Xsl عند استخدام الغرض XslTransform عند إجراء عمليات تحويل من جهة المخدم على وثائق XML.

كما يلزمنا استيراد فضاء أسماء System.Xml.Schema عند العمل مع الـــ Schemas (المخططات والهياكل).

أما بالنسبة لأغراض مثل XmlValidatingReader فلا نحتاج فيها إلى استيراد System.Xml.Schema.

في حال نسينا استيراد أي من فضاءات الأسماء اللازمة سنحصل على رسالة خطأ من الشكل:

Server Error in '/7035' Application.

Compilation Error

Description: An error occurred during the compilation of a resource required to service this request. Please review the following specific error details and modify your source code appropriately.

Compiler Error Message: BC30002: Type 'SqlConnection' is not defined.

Source Error:

```
Line 43:
Line 44:        'create a new Connection object using the connection string
Line 45:        Dim objConnect As New SqlConnection(strConnect)
Line 46:
Line 47:        'open the connection to the database
```

Source File: C:\Inetpub\wwwroot\7035\data-access\data01\datareader-sql.aspx **Line:** 45

Show Detailed Compiler Output:

Show Complete Compilation Source:

Version Information: Microsoft .NET Framework Version:1.0.3617.0; ASP.NET Version:1.0.3617.0

أغراض ADO.NET الأساسية (1)

اعتمدت طريقة الوصول التقليدية إلى البيانات –والتي استخدمت تقنية ADO– على غرض رئيسي وحيد هو Recordset، وكانت التقنية تتلخص في:

- تأسيس اتصال بقاعدة البيانات باستخدام مزود OLE-DB أو ODBC عبر OLE-DB
- تنفيذ أوامر على الاتصال المنشأ
- تخزين البيانات ضمن غرض RecordSet لاستعادتها
- يمكن لهذا السيناريو أن يتم باستخدام الغرض Command أو عن طريق غرض Connection مباشرة
- لإدراج أو تعديل البيانات يمكن ببساطة تنفيذ عبارة SQL أو إجرائية مخزنة باستخدام الغرض Connection والغرض Command دون استخدام الغرض Recordset

يكمن الفرق الرئيسي في أن الغرض DataReader يساعد في الوصول إلى البيانات باتجاه واحد و للقراءة فقط.

في حين يوفر الغرض Dataset آلية للتعامل مع أكثر من مجموعة من الصفوف من نفس مصدر البيانات، حيث يمكننا إنشاء غرض

DataSet من بيانات موجودة ضمن مصدر البيانات، أو من ملئها بصورة مباشرة صف تلو الآخر باستخدام النص البرمجي.

يحتوي كل جدول ضمن الغرض DataSet على معلومات حول القيم الأصلية للبيانات أثناء عملنا عليها، بحيث يمكن إرسال أي تعديلات على البيانات إلى مخزن البيانات في وقت لاحق.
يحتوي الغرض DataSet معلومات تصف محتوى الجداول، كأنماط الأعمدة، القواعد، والمفاتيح.
يجب أن نتذكر دائماً أن التركيز في الغرض DataSet هو القدرة على العمل بصورة دقيقة و فعالة في بيئة غير متصلة
يحافظ الغرض DataSet على محتوياته ويستطيع تحميل معلومات من وثيقة XML الحاوية على بيانات مهيكلة بالتنسيق الصحيح.

يعتمد الوصول إلى البيانات في NET. على نفس الخطوط العريضة ولكن باستخدام مجموعة أخرى من الأغراض. قد تبدو هذه الأغراض مشابهة لكنها مختلفة بشكل جذري داخلياً مع اختلاف في الأداء ومرونة أكبر. إذ تعتمد أغراض الوصول إلى البيانات في NET. على غرضين أساسيين الأول DataReader والثاني هو DataSet. ينفذ كلا الغرضين العمل الذي كان الغرض Recordset يقوم به.

أغراض ADO.NET الأساسية (2)

أهم أغراض ADO.NET الأساسية هي:

- أغراض Connection
- أغراض الأوامر
- أغراض DataAdapter
- الغرض DataSet
- الغرض DataView
- الغرض DataReader

أ – أغراض Connection

يشبه هذا الغرض بصورة كبيرة الغرض المستخدم مع ADO ويحوي خصائص مشابهة له. يستخدم هذا الغرض لوصل غرض Command بمخزن البيانات.

- يُستخدم الغرض OleDbConnection مع مزود OLE_DB
- يستخدم الغرض SqlConnection ما يسمى TDS مع نظام إدارة قواعد البيانات SQLServer
- كان من الممكن سابقاً لغرض Connection تنفيذ تعليمات SQL على مصدر البيانات أو فتح غرض RecordSet. أما في ASP.NET فهذا غير ممكن. على كل الأحوال، تؤمن أغراض الاتصال السابقة الوصول إلى المناقلات التي تكون قيد التنفيذ على مخزن بيانات معين

من أهم الأغراض المستعملة لكل من غرضي الاتصال OleDbConnect و SqlConnection:

الطريقة	الوصف
Open	تقوم بفتح اتصال إلى مصدر البيانات باستخدام الإعدادات الحالية مثل

	التي تحدد معلومات الاتصال المطلوب استخدامه. ConnectionString
Close	تقوم بإغلاق الاتصال الحالي مع مصدر البيانات.
BeginTransaction	تقوم ببدأ مناقلة مع مصدر البيانات و تعيد غرض Transaction الذي يمكن استخدامه لإتمام أو إلغاء المناقلة.

```
' SQL Server Provider Sample
Dim connStr As String = "data source=(local);" & _
       "initial catalog=Northwind;user id=sa"
Dim cnn As New SqlConnection(connStr)
cnn.Open()
' Use the connection in here…
If cnn.State = ConnectionState.Open Then
       cnn.Close()
End If
```

ب - أغراض الأوامر

- تشبه هذه الأغراض مثيلاتها Command في ADO وتمتلك نفس الخصائص، وتُستخدم لوصل غرض Connection مع غرض DataReader وغرض DataAdapter
- يستخدم الغرض OleDCommand مع مزود OLE-DB
- يستخدم الغرض SqlCommand مع خدمات البيانات الجدولية في SQLServer

يسمح الغرض Command بتنفيذ عبارات SQL أو إجرائيات مخزنة على مصدر البيانات. يتضمن هذا إعادة مجموعة صفوف (حيث نستخدم للوصول إليها غرض آخر كالغرض DataReader أو كالغرض DataAdapter) أو إعادة قيمة وحيدة، أو إعادة عدد العناصر المتأثرة بالاستعلامات التي لا تعيد مجموعة صفوف

الطريقة	الوصف
ExecuteNonQuery	تقوم بتنفيذ الأوامر المعرفة في الخاصة CommandText باستخدام الاتصال المعرف بالخاصة Connection والمرتبطة باستعلام لا يعيد أي صفوف (كتعليمة Update و Delete أو Insert). تجري إعادة رقم صحيح يعبر عن عدد الصفوف التي تأثرت بهذا الاستعلام.
ExecuteReader	تقوم بتنفيذ الأمر المعرف في الخاصة CommandText باستخدام الاتصال المعرف في الخاصة Connection. تعيد هذه الطريقة غرض reader متصل مع مجموعة الصفوف في قاعدة المعطيات بشكل يسمح بالحصول على الصفوف.
ExecuteScalar	تقوم بتنفيذ الأمر المعرف في الخاصة CommandText باستخدام الاتصال المحدد في الخاصة Connection. يعيد قيمة وحيدة هي العمود الأول من السطر الأول من مجموعة الصفوف التي يعيدها الاستعلام، ويتم إهمال جميع القيم الأخرى المعادة. هذه الطريقة سريعة و فعالة عندما يكون المطلوب إعادة قيمة وحيدة فقط.

مثال:

```
' OLE DB Provider Sample
Dim strSQL As String = "SELECT * FROM customer"
Dim cmd As New OleDbCommand(strSQL ,cnn)
```

```
' SQL Server Provider Sample
Dim strSQL As String = "SELECT * FROM customers"
Dim cmd As New SqlCommand(strSQL ,cnn)
```

ج - أغراض DataAdapter

يعتبر هذا النوع من الأغراض نوعاً جديداً يقوم بوصل غرض Command أو أكثر بغرض DataSet.

- الغرض OleDbDataAdapter المستخدم مع مزود OLE-DB
- الغرض SqlDataAdapter المستخدم مع خدمات البيانات الجدولية الخاصة ب MS SQl Server

توفر هذه الأغراض أربعة خصائص تُعرِّف الأوامر المستخدمة للتعامل مع البيانات في مخزن البيانات:

SelectCommand و InsertCommand و UpdateCommand و DeleteCommand

حيث تشكل كل من هذه الخصائص مؤشر إلى غرض Command (يمكن لكل هذه الأغراض الاشتراك بغرض Connection وحيد).

تقدم كل من أغراض OleDbDataAdapter و SqlDataAdapter مجموعة من الطرق للعمل مع أغراض DataSet التي تطبق عليها. أهم تلك الطرق ثلاثة هي:

الطريقة	الوصف
Fill	تقوم بتنفيذ الأمر SelectCommand لتأهيل الغرض DataSet بالبيانات من مصدر البيانات. كما يمكن أيضاً استخدامها لتحديث المعلومات المتوفرة ضمن جدول في غرض DataSet بالتعديلات التي تمت على البيانات في مصدر البيانات الأصلي إذا كان هناك مفتاح رئيسي يميز صفوف البيانات في الجدول ضمن غرض Dataset.
FillSchema	تستخدم الأمر SelectCommand لاستخلاص البنية الهيكلية لجدول من مصدر بيانات و تقوم بإنشاء جدول فارغ في غرض DataSet مع جميع القيود التي تحددها تلك البنية.
Update	تقوم باستدعاء الأوامر InsertCommand، UpdateCommand، DeleteCommand، لكل صف تم تعديله أو حذفه من الغرض DataSet بحيث يجري تحديث هذه البيانات ضمن مصدر البيانات الأساسي، باستخدام التعديلات التي تمت على محتوى الغرض DataSet. تشبه هذه الطريقة، الطريقة UpdateBatch المستخدمة مع غرض Recordset في ADO و لكن في حالة ADO لا يتم التعديل على أكثر من جدول.

مثال:

```
Dim connStr As String = "Provider=VFPOLEDB.1;Data Source=" & _
       "C:\SAMPLES\DATA\TESTDATA.DBC"

Dim strSQL As String = "SELECT * FROM Products"

Dim oda As New OleDbDataAdapter(strSQL, connStr)

Dim cmdInsert As New OleDbCommand("INSERT INTO Products" & _
     "(product_id, prod_name) VALUES (10,'car')")

oda.InsertCommand = cmdInsert
```

د - الغرض DataSet

يقدم الغرض DataSet أساسيات التعامل مع قواعد البيانات العلائقية ووسائل تخزين البيانات في بيئة غير متصلة.

تتلخص الفروق الأساسية بين الغرض DataSet و الغرض RecordSet (في ADO) بما يلي:

- يمكن لغرض DataSet أن يتعامل مع أكثر من مجموعة صفوف ويحتوي معلومات تخص العلاقة بينها، في حين لا يقدم الغرض RecordSet سوى إمكانية التعامل مع مجموعة صفوف واحدة
- يوفر الغرض DataSet وصول غير متصل إلى البيانات بصورة آلية
- يكون كل جدول في الغرض DataSet عبارة عن غرض من النوع DataTable ينتمي إلى المجموعة Tables.
- يحتوي كل غرض DataTable على مجموعة من أغراض DataRow و مجموعة من أغراض DataColumn
- هناك أيضاً مجموعات خاصة بالقيود كقيد المفتاح الأساسي و القيم التلقائية و العلاقات بين الجداول

أخيراً لدينا الغرض DefaultView الخاص بكل جدول و المخصص لإنشاء غرض DataView على ذلك الجدول، وذلك لتقديم إمكانيات البحث في البيانات و التعامل معها أو ربطها بعنصر تحكم.

نقوم بتأهيل هذا الغرض من مخزن بيانات، ثم نقوم بالتعامل معه في حالة غير متصلة مع هذا المخزن، ثم نقوم بالاتصال وإتمام التغييرات على مخزن البيانات حسب الحاجة.

يكون كل جدول في الغرض DataSet عبارة عن غرض من النوع DataTable ينتمي إلى المجموعة Tables.

يقدم الغرض DataSet مجموعة من الطرق التي يمكن استخدامها للتعامل مع محتويات الجداول أو للتعامل مع العلاقات القائمة فيما بينها، مثل عمليات مسح محتوى غرض DataSet أو عملية دمج محتويات أكثر من غرض DataSet:

الطريقة	الوصف
Clear	تقوم بإزالة جميع البيانات المخزنة في الغرض DataSet و ذلك بتفريغ جميع الجدوال التي تحتويها. تكون عملية تدمير الغرض و إعادة إنشائه أكثر فعالية، في بعض الأحيان) إلا في حال الرغبة بالمحافظة على مؤشر لهذا العنصر.

Merge	تقوم هذه الطريقة بدمج محتوى غرض DataSet مع محتوى غرض DataSet آخر منتجةً غرض DataSet يحتوي جميع البيانات من كلا الغرضين.

ذكرنا سابقاً التنسيق التلقائي الذي تعتمده NET. هو تنسيق Xml عند تخزين البيانات. لذلك يقدم الغرض DataSet مجموعة من الطرق لقراءة و كتابة بيانات من وإلى وثائق XML.

الطريقة	الوصف
ReadXml و ReadXmlSchema	تقوم بقراءة معلومات وثيقة XML أو هيكل XML و تأهيل غرض DataSet بها.
GetXml و GetXmlSchema	تعيد سلسة محرفية تحتوي وثيقة XML أو هيكل XML الذي يمثل البيانات في غرض DataSet.
WriteXml و WriteXmlSchem	تقوم بكتابة وثيقة XML أو هيكل XML الذي يمثل البيانات ضمن غرض DataSet إلى ملف على القرص أو غرض Reader/Writer

تقوم أغراض DataSet مع أغراض DataTable التي تحتويها بالاحتفاظ بسجل عن قيم محتوياتها عند إنشاءها أو تحميلها إذ يعتبر هذا الأمر مطلب أساسي مهم للسماح بتثبيت التغيرات في مخزن البيانات الرئيسي وبالأخص في بيئة متعددة المستخدمين.

مثال

```
Dim connStr As String = "Provider=VFPOLEDB.1;Data Source=" & _
       "C:\SAMPLES\DATA\TESTDATA.DBC"

Dim strSQL As String = "SELECT * FROM Products"
Dim oda As New OleDbDataAdapter(strSQL, connStr)
Dim ds As New DataSet()
Dim dr As DataRow

oda.Fill(ds, "ProductInfo")

For Each dr In ds.Tables("ProductInfo").Rows
  lstDemo.Items.Add(dr("Prod_Name").ToString)
Next dr

' Want to bind a Windows Form grid? It's this easy:
dgdDemo.DataSource = ds.Tables("ProductInfo")
```

يقدم الغرض DataSet أربعة طرائق لإعطاء قدرة أكبر على التحكم في كيفية ولحظة تثبيت البيانات:

الطريقة	الوصف
AcceptChanges	تقوم بإتمام وتثبيت جميع التغييرات الحاصلة على الجداول والعلاقات ضمن غرض DataSet منذ أن تم تحميلها أو منذ آخر مرة تم تنفيذ الطريقة AcceptChanges فيها.
GetChanges	تعيد هذه الطريقة غرض DataSet يحتوي بعض أو كل التغيرات التي تمت منذ تم

	تحميل الغرض أو منذ آخر مرة تم فيها تنفيذ الطريقة AcceptChanges.
HasChanges	تحدد فيما إذا تمت أي تعديلات على البيانات منذ تحميل الغرض أو منذ آخر مرة تم تنفيذ الطريقة AcceptChanges فيها.
RejectChanges	تقوم بإهمال جميع التعديلات التي تمت على القيم في الجداول ضمن غرض DataSet منذ تحميل الغرض أو منذ آخر مرة تم تنفيذ الطريقة AcceptChanges فيها. تعيد هذه الطريقة المعلومات إلى الوضع الأصلي و تزيل جميع المعلومات المخزنة عن التغييرات.

الغرض DataTable:

كل غرض DataTable يمتلك خاصة تسمى Rows تؤشر إلى غرض DataRowCollection وهو عبارة عن مجموعة أغراض DataRow.

يوفر الغرض DataTable مجموعة من الخصائص و الطرق التي تسمح بالتعامل مع كل جدول من جداول غرض DataSet على حدى. من أكثر الطرق استخداماً هي الطرق Clear و AcceptChanges و RegectChanges و هي مطابقة لتلك التي يدعمها الغرض DataSet إلا أنها تطبق فقط على جدول وحيد هو الجدول الذي يؤشر إليه الغرض DataTable. كما أن هناك مجموعة من الطرق الأخرى التي تسمح بالتعامل مع محتويات الجدول وهي:

الطريقة	الوصف
NewRow	تقوم بإنشاء صف في الجدول.حيث يتم إدخال القيم إلى الصف باستخدام النص البرمجي .
Select	تقوم بإعادة مجموعة من الأسطر التي تطابق تعبير تصفية معين.

مثال:

```
<%@ Page Language="vb" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

    Sub Page_Load(sender As Object, e As EventArgs)
    'declarations
    Dim mycon As OleDbConnection
    Dim mycmd As OleDbCommand
    Dim mydap As OleDbDataAdapter
    Dim mydst As DataSet

    mycon = New OleDbConnection( _
                  "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                      "Data Source=c:\myDb.mdb")

    mydap = New OleDbDataAdapter ( _
               "select col1,col2 from myTable", _
                  mycon)
    mydst = New DataSet()
    mydap.Fill(mydst,"col2")
```

```
    mygrid.DataSource = mydst.Tables("col2")
    mygrid.DataBind

    End Sub

</script>
<html>
<head>
<title>Using DataAdapter - in  dot net</title>

</head>

<body bgcolor="#FFFFFF" text="#000000">
<asp:DataGrid id="mygrid"
    Font-Name="Arial" Font-Size="10pt" BackColor="#CCCCCC"
    Width="90%" Border="0" Cellpadding="5"
    runat="server" AutoGenerateColumns="False">
<HeaderStyle font-bold="True" HorizontalAlign="Center"/>
<Columns>
    <asp:BoundColumn DataField="col1" HeaderText="col1" />
    <asp:BoundColumn DataField="col2" HeaderText="col2" />
    />
</Columns>
</asp:DataGrid>

</body>
</html>
```

الغرض DataTable:

يمكن الوصول إلى الجداول المحتواة في غرض DataSet عن طريق الغرض DataTable

الطرق الأساسية لغرض DataRowCollection:

يوفر غرض DataRowCollection مجموعة من الطرق لإضافة أو إزالة صفوف ولإيجاد صف بناء (Constructor) على قيمة المفتاح الأساسي للجدول.

الطرق الأساسية لغرض DataRowCollection

الطريقة	الوصف
Add	تقوم هذه الطريقة بإضافة سجل جديد تم إنشاؤه بالطريقة NewRow (الخاص بـــ DataTable) إلى الجدول
Remove	تقوم بإزالة أغراض DataRow المحددة من الجدول
RemoveAt	تقوم بإزالة سطر محدد بقيمة دليل موقعه في الجدول
Find	تقوم هذه الطريق بأخذ مصفوفة من قيم المفتاح الرئيسي و تعيد الصفوف المقابلة لها في الجدول كأغراض DataRow

الطرق الأساسية لغرض DataRow:

هذا الغرض يمثل الصف بحد ذاته ضمن الجدول و ضمن الغرض DataRowCollection.

يمتلك هذا الغرض طرق AcceptChanges و RejectChanges والتي تقوم بنفس العمل الذي تقوم به مثيلاتها في الغرض DataTable. إضافة إلى هذه الطرق يقدم هذا الغرض مجموعة من الطرق المستخدمة للتعامل مع البيانات في صف واحد من الجدول.

الطريقة	الوصف
BeginEdit, EndEdit, CancelEdit	تستخدم لتحويل الصف إلى وضعية التحرير وحفظ أو إلغاء التغييرات على البيانات في وضع التحرير.
Delete	تقوم بتمييز الصف كصف محذوف، و لكن لن تتم إزالته من الجدول لحين تنفيذ الطريقة AcceptChanges أو Update.
GetChildsRows	تقوم هذه الطريقة بإعادة مجموعة من الصفوف من جدول مرتبط بهذا الصف بعلاقة ابن.
SetColumnError وGetColumnsInError	تستخدم لتعيين و إعادة حالة الخطأ لهذا الصف. حيث تستخدم هذه الطرق مع الخصائص HasErrors و RowError لتحري أو لتحديد أماكن الخطأ.

يقوم هذا المثال باستخدام الطريقة Find لإيجاد صف يحتوي قيمة محددة ثم يقوم بحذفه.

```
Private Sub RemoveFoundRow(ByVal table As DataTable)
    Dim rowCollection As DataRowCollection = table.Rows

    ' Test to see if the collection contains the value.
    If rowCollection.Contains(TextBox1.Text) Then
        Dim foundRow As DataRow = rowCollection.Find(TextBox1.Text)
        rowCollection.Remove(foundRow)
        Console.WriteLine("Row Deleted")
    Else
        Console.WriteLine("No such row found.")
    End If
 End Sub
```

الطرق الأساسية لغرض DataRowCollection:

يمثل هذا الغرض كما ذكرنا مجموعة من الصفوف ضمن غرض DataTable تم تحديدها بالخاصة Rows لهذا الغرض.

الطرق الأساسية لغرض DataRow:

هذا الغرض يمثل الصف بحد ذاته ضمن الجدول وضمن الغرض DataRowCollection.

ه - الغرض DataView

يمكننا تأهيل غرض DataView من خلال جدول في غرض DataSet.

يقوم الغرض DataView بالتعامل مع جدول أو مجموعة من الصفوف في جدول. يمكن إنشاءه باستخدام الغرض DefaultView الخاص بالجدول أو من خلال الغرض DataTable الذي يقوم باختيار مجموعة جزئية من الصفوف في الجدول.

بصورة عامة إن أفضل طريقة للتعامل مع محتويات الجدول ضمن الغرض DataSet هي إنشاء غرض DataView من الجدول المراد استخدامه ثم استعمال الطرق التي يوفرها:

الطريقة	الوصف
AddNew	تقوم هذه الطريقة بإضافة صف جديد إلى الغرض DataView. يمكن بعدها استخدام النص البرمجي لإدراج قيم في هذا الصف.
Delete	تقوم هذه الطريقة بإزالة الصف محدد من الغرض DataView.
`Find	تأخذ هذه الطريقة كمعامل قيمة وحيدة أو مصفوفة من القيم و تعيد الدليل للصف الذي يطابق هذه القيم.
FindRows	تأخذ قيمة وحيدة أو مصفوفة من القيم و تعيد مجموعة من الأغراض DataRow التي تطابق هذه القيم.

مثال:

```
Dim strSQL As String = "SELECT * FROM Products"
Dim sda As New SqlDataAdapter(strSQL, connStr)
Dim ds As New DataSet()

sda.Fill(ds, "ProductInfo")

Dim dv As New DataView(ds.Tables(“ProductInfo”))

dv.Sort = “Prod_Name ASC”

dv.RowFilter = “In_Stock > 100”

‘ Bind the DataGrid to this DataView.
Me.dgdDemo.DataSource = dv
```

كما ذكرنا مسبقاً يمكننا تأهيل غرض DataView من خلال جدول في غرض DataSet.

يقوم الغرض DataView بالتعامل مع جدول أو مجموعة من الصفوف في جدول.

و - الغرض DataReader

توفر أغراض DataSet أرضية مناسبة للوصول إلى البيانات في بيئة غير متصلة، ولكن في الكثير من الحالات، وبالأخص في تلك التي نحتاج فيها إلى طريقة سريعة وفعالة للوصول إلى البيانات بدون الاضطرار إلى سحب جميع البيانات،

يعد استخدام الغرض DataReader من أنجع الحلول لتحقيق:

- سحب سجل أو أكثر أو سحب قيم من حقول محددة
- تنفيذ تعبيرات SQL INSERT، UPDATE، DELETE
- عندما يكون لدينا فيها كمية كبيرة من البيانات أكبر من أن تتسع لها أغراض DataSet
- ربط عناصر التحكم من جهة المخدم

هناك نوعان أساسيان من الغرض DataReader:

- الغرض OleDbDataReader و هو يستخدم OLE-DB
- الغرض SqlDataReader الذي يستخدم خدمات البيانات الجدولية الخاصة بمخدم SQLServer

يقوم الغرض DataReader بتنفيذ تعليمة SQL أو تخزين إجرائيات مخزنة لاستحضار مجموعة من صفوف البيانات والتجوال فيما بينها إذ تتم المحافظة على الاتصال مع مخزن البيانات في ذلك الوقت.

- يوفر الغرض DataReader شبيه للمؤشرات المستخدمة مع مخازن البيانات والتي تستخدم تعبيرات SQL أو الإجرائيات المخزنة لاستخلاص مجموعة الصفوف
- يوفر الغرض DataReader إمكانية تنفيذ تعليمات SQL أو إجرائيات مخزنة لتحديث البيانات
- لا يوفر هذا الغرض وصول غير متصل إلى البيانات
- الوصول إلى مجموعة الصفوف التي يشير إليها الغرض DataReader هو وصول مخصص للقراءة فقط و باتجاه واحد

لاستخدام غرض DataReader نقوم بإنشاء غرض Command ثم نستخدمه لتنفيذ تعبيرات SQL أو إجرائيات مخزنة و إعادة غرض DataReader.

يمكننا بعدها الدوران عبر هذه الصفوف و الأعمدة باستخدام الغرض DataReader لاستخلاص النتائج من مخزن البيانات.

أكثر طرق الغرض DataReader استخداماً هي التالية:

الطريقة	الوصف
Read	يقوم بدفع مؤشر الصف خطوة إلى الأمام ليؤشر إلى السطر التالي بحيث يتم الوصول إلى قيم الأعمدة باستخدام اسم العمود أو رقم التسلسل الخاص به.
GetValue	تعيد قيمة من الصف الحالي بنفس التنسيق المُستخدم في مصدر البيانات وذلك بتحديد دليل العمود. لتنفيذ عملية بشكل أبسط و لكن أكثر فعالية يمكن استخدام اسم العمود مباشرة و ذلك بالشكل . Value=DataReader("col-Name")
GetValues	يقوم باعادة قيمة أو مجموعة قيم من الصف الحالي بتنسيقها الأصلي وذلك ضمن مصفوفة.
Getxxxxxx	تقوم بإعادة قيمة من السطر الحالي بتنسيق نمط البيانات المحدد حسب الطريقة xxxxxx. مثل GetBoolean,GetInt16,GetChars.
NextResult	تقوم هذه الطريقة بنقل مؤشر الصف إلى مجموعة الصفوف الأخرى عندما تقوم تعليمة SQL أو الإجرائية المخزنة بإعادة أكثر من مجموعة صفوف. تجدر الإشارة إلى أن هذه الطريقة ليست مشابهة للطريقة MoveNext لأنها لاتنتقل ضمن نفس مجموعة الصفوف بل من مجموعة صفوف إلى أخرى.

Close	تقوم بإغلاق الغرض DataReader و تحرير المؤشر الذي يشير إلى مجموعة الصفوف.

مثال:

```
SqlConnection conn = new SqlConnection(connectionString);
 SqlCommand comm = new SqlCommand("select * from mytable", conn);
 comm.Connection.Open();
 SqlDataReader r =
    comm.ExecuteReader(CommandBehavior.CloseConnection);
 while(r.Read())
 {
   Console.WriteLine(r.GetString(0));
 }
 r.Close();
 conn.Close()
```

توفر أغراض DataSet أرضية مناسبة للوصول إلى البيانات في بيئة غير متصلة، ولكن في الكثير من الحالات، وبالأخص في تلك التي نحتاج فيها إلى طريقة سريعة و فعالة للوصول إلى البيانات بدون الاضطرار إلى سحب جميع البيانات،
يعد استخدام الغرض DataReader من أنجع الحلول لتحقيق:

- سحب سجل أو أكثر أو سحب قيم من حقول محددة
- تنفيذ تعبيرات SQL INSERT، UPDATE،DELETE
- عندما يكون لدينا فيها كمية كبيرة من البيانات أكبر من أن تتسع لها أغراض DataSet

الوصول إلى مجموعة الصفوف التي يشير إليها الغرض DataReader هو وصول مخصص للقراءة فقط وباتجاه واحد.

متى يجب استخدام غرض DataReader ومتى يجب استخدام غرض DataSet:

- عند البدء ببناء تطبيقاتنا للوصول إلى مخازن البيانات، يجب علينا أن نفكر بنوع الوصول الذي نريد وكيف سيتم استعمال البيانات
- لا بد وأنه أصبح من الواضح مما ذكرناه مسبقاً أن أغراض DataSet تسبب حمل وتعقيد لا يمكن إهماله مقارنة مع الغرض DataReader من حيث الأداء واستهلاك الذاكرة. إذاً لا بد لنا من التركيز على استخدام الغرض DataReader عوضاً عن DataSet

هناك بعض الحالات التي لا يمكن فيها الاستغناء عن DataSet وهي:

- عند حاجتنا إلى استخدام البيانات بشكل غير متصل مع مخزن البيانات و تمرير هذه البيانات إلى أطر أخرى ضمن التطبيق، وتخزينها ، وتحريرها
- عند الحاجة إلى تخزين، ونقل، والوصول إلى أكثر من جدول (أكثر من غرض DataTable) ومعالجة العلاقات بين هذه الجداول
- عند الحاجة إلى تحديث البيانات في قاعدة البيانات باستخدام طرق خاصة بغرض DataSet و DataAdapter عوضاً عن استخدام تعبيرات SQL UPDATE أو استخدام الإجرائيات المخزنة
- عند الحاجة لإدارة الوضع بشكل أفضل في بيئة متعددة المستخدمين

- عندما نريد الاستفادة من المزامنة بين وثائق XML و مجموعة الصفوف العلائقية المقابلة
- في بعض حالات الربط مع عناصر التحكم مثل حالات الربط مع أكثر من عنصر في نفس الوقت، أو تقسيم السجلات إلى صفحات في عنصر تحكم DataGrid، إذ نستخدم في هذه الحالة الغرض DataView المنشأ من جدول في غرض DataSet
- في حال أردنا الدوران ضمن صفوف البيانات وأردنا الحصول على حرية في اتجاه الحركة للأمام أو الخلف. هنا أيضاً لا يمكننا استخدام الغرض DataReader

متى يجب استخدام غرض DataReader ومتى يجب استخدام غرض DataSet:

- عند البدء ببناء تطبيقاتنا للوصول إلى مخازن البيانات، يجب علينا أن نفكر بنوع الوصول الذي نريد وكيف سيتم استعمال البيانات
- لا بد وأنه أصبح من الواضح مما ذكرناه مسبقاً أن أغراض DataSet تسبب حمل وتعقيد لا يمكن إهماله مقارنة مع الغرض DataReader من حيث الأداء و استهلاك الذاكرة. إذاً لابد لنا من التركيز على استخدام الغرض DataReader عوضاً عن DataSet

مزودات البيانات العلائقية في NET

كما رأينا سابقاً تستخدم NET. مزودات بيانات للاتصال بمخازن البيانات. يوفر إطار عمل NET. دعم لمجموعة المزودات التالية:

اسم المزود	الوصف
SQLOLEDB	مزود OLE-DB لـــ SQLServer
MSDAORA	مزود OLE-DB لـــ ORACLE
Microsoft.Jet.OLEDB.4.0	مزود OLE-DB لـــ Access مصادر البيانات التي تستخدم مزود Jet

تقدم NET. أيضاً المزود ODBC الذي تم تطويره ليسمح بالاتصال بالعديد من التجهيزات و مخازن البيانات، وهو يستخدم أغراض مشابهة لتلك المستخدمة مع المزودات الأخرى فهو يتضمن مثلاً OdbcConnection، OdbcCommand، OdbcDataReader، OdbcDataAdapter...إلخ.

تقدم NET. أيضاً المزود ODBC الذي تم تطويره ليسمح بالاتصال بالعديد من التجهيزات ومخازن البيانات، وهو يستخدم أغراض مشابهة لتلك المستخدمة مع المزودات الأخرى.

أمثلة

```
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
```

```
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
end sub
</script>
```

```
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
end sub
</script>
```

```
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm,dbread
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
dbread=dbcomm.ExecuteReader()
end sub
</script>
```

```
<%@Page Language="VB"%>


<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb" %>


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>The .NET DataSet and OleDbDataAdapter Objects</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">The .NET DataSet and OleDbDataAdapter
Objects</span><hr />
<!-----------------------------------------------------------------
------>


<div>Connection string: <b><span id="outConnect"
runat="server"></span></b></div>
<div>SELECT command: <b><span id="outSelect"
```

```
runat="server"></span></b></div>
<div id="outError" runat="server"> </div>

<asp:datagrid id="dgrResult" runat="server" />

<script language="vb" runat="server">

Sub Page_Load()

   'get connection string from web.config file
   Dim strConnect As String
   strConnect = ConfigurationSettings.AppSettings("DsnWroxBooksJet") _
              & Request.PhysicalApplicationPath _
              & "databases\WroxBooks.mdb"
   outConnect.innerText = strConnect 'and display it

   'specify the SELECT statement to extract the data
   Dim strSelect As String
   strSelect = "SELECT * FROM BookList WHERE ISBN LIKE '07645437%'"
   outSelect.innerText = strSelect   'and display it

   'declare a variable to hold a DataSet object
   'note that we have to create it outside the Try..Catch block
   'as this is a separate block and so is a different scope
   Dim objDataSet As New DataSet()

   Try

      'create a new Connection object using the connection string
      Dim objConnect As New OleDbConnection(strConnect)

      'create a new DataAdapter using the connection object and select
statement
      Dim objDataAdapter As New OleDbDataAdapter(strSelect, objConnect)

      'fill the dataset with data from the DataAdapter object
      objDataAdapter.Fill(objDataSet, "Books")

   Catch objError As Exception

      'display error details
      outError.innerHTML = "<b>* Error while accessing data</b>.<br />"
_
          & objError.Message & "<br />" & objError.Source
      Exit Sub  ' and stop execution

```

```
   End Try

   'create a DataView object for the Books table in the DataSet
   Dim objDataView As New DataView(objDataSet.Tables("Books"))

   'assign the DataView object to the DataGrid control
   dgrResult.DataSource = objDataView
   dgrResult.DataBind()   'and bind (display) the data

End Sub
</script>

<!-------------------------------------------------------------------
------>
<!-- #include file="..\global\foot.inc" -->
</body>
</html>
```

```
Dim ds As New DataSet()

Dim strSQL As String = "SELECT Cust_ID, Order_Id, “ & _
     “Order_Date FROM Orders WHERE Year(Order_Date) > 1997"
Dim oda As New OleDbDataAdapter(strSQL, connStr)
oda.Fill(ds, "OrderInfo")

strSQL = "SELECT CustomerID FROM Customers “  & _
     “ORDER BY CustomerID"
Dim sda As New SqlDataAdapter(strSQL, connStr)
sda.Fill(ds, "CustomerInfo")

Dim dc1 As DataColumn = _
     ds.Tables("CustomerInfo").Columns("CustomerId")
Dim dc2 As DataColumn = _
     ds.Tables("OrderInfo").Columns("Cust_Id")

Dim dr As New DataRelation("CustomersToOrders", dc1, dc2)
ds.Relations.Add(dr)

Dim drCustomer As DataRow
Dim drOrder As DataRow

For Each drCustomer In ds.Tables("CustomerInfo").Rows
     lstDemo.Items.Add("Customer: " & _
     drCustomer("CustomerId").ToString())
     ' Iterate through related rows.
     For Each drOrder In drCustomer.GetChildRows(dr)
```

```
            lstDemo.Items.Add( _
                String.Format("   Order {0} placed on {1:d}", _
                drOrder("Order_ID"), drOrder("Order_Date")))
        Next drOrder
    Next drCustomer
```

مقدمة عن XML في NET

تكلمنا سابقاً عن المزايا الجديدة في إطار العمل NET. والمخصصة للوصول إلى البيانات العلائقية وقمنا بمقارنة بسيطة مع التقنيات التقليدية التي كانت مستخدمة مع ADO.

أصبحت لغة XML وبسرعة، اللغة الأساسية على الوب وتم اعتمادها من قبل العديد من التطبيقات. لذا سنركز في هذا الجزء على الطريقة التي تدعم بها NET. لغة XML.

أغراض XML الأساسية في (NET.):

قدمت منظمة W3C مجموعة من المعايير التي حددت البنية والواجهات التي يجب تزويد التطبيقات بها للوصول إلى وثائق XML. تسمى هذه المعايير بـــ (DOM).

وقد تبنت NET. دعم هذه المعايير من خلال الأغراض XmlDocument و XmlDataDocument.

توفر هذه الأغراض دعماً كاملاً لمعايير (XML DOM) من الدرجة الثانية.
وسعت NET. دعمها لـــ XML عبر تقديم تقنيات للتعامل مع وثائق XML، وهياكل XML وملفات الأنماط.

تتوزع أغراض XML الأساسية في ثلاث مجموعات:

المجموعة الأولى لقراءة وكتابة وتحويل ملفات XML، وتتضمن هذه المجموعة أغراض XmlTextReader و XmlTextWriter و XmlNodeReader إضافة إلى غرض XslTransform الخاص بإنشاء ملف بتنسيق مختلف عن وثيقة XML الأصلية.

المجموعة الثانية خاصة بتخزين وثائق XML والتعامل معها وهي تتضمن الأغراض مثل XmlDocument، XmlDataDocument، XPathDocument.

المجموعة الثالثة خاصة باستعلام XML و نستخدم فيها الغرض XPathNavigator.

سنلاحظ تداخل بين هذه الوظائف. فلاختبار وثيقة XML أثناء قراءتها، نستخدم الغرض XmlValidatingReader و هناك أغراض أخرى لإنشاء و تحرير هياكل و مخططات XML. كما يمكننا أيضاً استخدام الغرض XslTransform لتنفيذ استعلام على وثيقة

بالإضافة إلى تحويلها إلى تنسيق مختلف.

وسعت NET. دعمها لـــ XML عبر تقديم تقنيات للتعامل مع وثائق XML، وهياكل XML وملفات الأنماط.

أغراض Document

لدينا ثلاث آليات تطبيقية لغرض Document:

- الغرض XmlDocument : هو عبارة عن آلية تطبيق Net. لمعايير Dom. تتضمن خصائص وطرق هذا الغرض تلك المعرفة من قبل W3C للتعامل مع وثائق XML مع بعض الإضافات لجعل العمليات الأساسية أسهل.
- الغرض XmlDataDocument : هو عبارة عن امتداد للغرض XmlDocument يوفر نفس مجموعة الخصائص والطرق، كما يعمل كجسر وصل بين XML وطرائق قواعد البيانات العلائقية.
- الغرض XPathDocumen : يمثل آلية سريعة لتخزين وثائق XML، وهو مصمم خصيصاً ليتم الوصول إليه باستخدام الغرض XPathNavigator، باستعمال استعلامات XPath فقط أو الملاحة ضمن الوثيقة عنصر تلو الآخر باستخدام تقنية "Pull".

الطرق الأساسية لأغراض Document:

الطريقة	الوصف
Createxxxxxx	تقوم بإنشاء عقدة في وثيقة XML اعتماداً على اسم الطريقة، مثل CreateElement، CreateComment، CreateTextNode...إلخ
CloneNode	تقوم بإنشاء نسخة مثيلة عن عقدة XML مثل نسخة من عقدة Element
GetElementById	تقوم بإعادة عقدة وحيدة باستخدام قيمة الواصفة ID كمعامل.
GetElementsByTagname	تقوم بإعادة عقدة وحيدة باستخدام اسم التأشيرة كمعامل

لا يمتلك الغرض XPathDocument طرقاً عامة ذات فائدة غير الطريقة CreateNavigator كونه مصمم ليعمل فقط مع الغرض XPathNavigator

في حين يوفر النوعان الآخران لغرض Document مجموعة كاملة من الطرق والخصائص وهي تلك المحددة في معايير W3C DOM

إضافة لما ذكر يوفر هذان النوعان مجموعة مفيدة من الطرق والخصائص التي تستخدم بشكل كبير للعمل مع وثائق XML وهي تتضمن طرق لإنشاء أنواع معينة من العقد والوصول إلى عقد موجودة

هناك أيضاً مجموعة من الطرق لتحميل وحفظ XML من وإلى غرض وثيقة XML.

Save	تقوم بحفظ وثيقة XML كاملة إلى ملف على القرص أو إلى غرض Stream أو إلىغرض XmlTextWriter.
ReadNode	تقوم بتحميل عقدة من وثيقة XML يجري التأشير إليها بغرض XmlTextReader أو غرض

	.XmlNodeReader
WriteTo	تقوم بكتابة عقدة إلى وثيقة XML يجري التأشير إليها بغرض XmlTextWriter
WriteContentTo	تقوم بكتابة عقدة وكل ما يندرج تحتها إلى وثيقة XML أخرى يجري التأشير إليها بغرض .XmlTextWrite

أما إذا أردنا استعمال الغرض XPathNavigator في وثيقتنا فإننا نقوم بإنشاءه باستخدام الطريقة CreateNavigator:

الطريقة	الوصف
CreateNavigator	تقوم بإنشاء و إعادة غرض XPathNavigator مبني على وثيقة XMLالحالية. يمكن تطبيقه على جميع أنواع أغراض Document . و مع XmlDocument و XmlDataDocument تقبل هذه الطريقة معامل إضافي يشكل مؤشر إلى عقدة في الوثيقة و التي ستكون موقع البداية لغرض XPathNavigator

يدعم غرض XmlDataDocument خاصة إضافية عن الخصائص يقدمها الغرض XmlDocument :

الخاصة	الوصف
DataSet	تقوم بإعادة محتوى وثيقة XML كغرض DataSet.

كذلك يدعم الغرض XmlDataDocument طريقتين إضافيتين توفران وصول أقوى إلى محتوى الوثيقة و ذلك بالتعامل معها كمجموعة صفوف أو جدول بيانات.

الطريقة	الوصف
GetRowFromElement	يعيد غرض DataRow يمثل العناصر في الوثيقة.
GetElementFromRow	يعيد غرض XmlElement يمثل DataRow في جدول ضمن DataSet

الغرض XPathNavigator

- لجعل العمل مع وثائق XML أسهل تم تعريف الغرض XPathNavigator في فضاء الأسماء System.Xml والذي يمكن استخدامه للتجوال ضمن وثيقة XML أو للاستعلام عن مكون ضمن الوثيقة باستخدام تعبير XPath
- نشير إلى أنه يمكننا استخدام الغرض XPathNavigator مع أي غرض وثيقة XML وليس فقط مع XPathDocument. أي يمكننا إنشاء غرض XPathNavigator مبني على غرض XMLDocument أو على غرض XmlDataDocument
- يقدم الغرض XPathNavigator مجموعة من الطرق و الخصائص التي تسمح بالتجول ضمن وثيقة XML كالانتقال بين العقد بالترتيب أو بتجاوز العقد حتى الوصول إلى عقدة من نمط معين
- يقدم الغرض XPathNavigator مجموعة من الطرق التي تقبل تعبيرات XPath، أواسم العقدة، أونمط العقدة، وتعيد العقدة أو مجموعة العقد المطابقة. عندها نستطيع التجوال عبر هذه العقد

يمكن إنشاء الغرض XPathNavigator فقط من غرض وثيقة موجود و ذلك كما يلي:

```
Dim objNav1 As XPathNavigator = objXMLDoc.CreateNavigator()
```

```
Dim objNav2 As XPathNavigator = objXMLDataDoc.CreateNavigator()
Dim objNav3 As XPathNavigator = objXPathDoc.CreateNavigator()
```

- يسمح هذا الغرض بالتجوال عبر الوثيقة واختيار عقدة ضمن وثيقة XML والوصول إليها
- يمكننا إنشاء أكثر من غرض من هذا النوع على نفس الوثيقة ومقارنة مواقعها
- لتحرير وثيقة XML نستخدم مؤشراً إلى العقدة الحالية لهذا الغرض أو غرض XPathNodeIterator الذي يحتوي أكثر من مجموعة من العقد، ونقوم باستدعاء الطرق الخاصة بعقدة ما من المجموعة
- في نفس الوقت يوفر الغرض XPathNavigator تفاصيل حول العقدة الحالية وبهذا يكون لدينا طريقتان للوصول إلى معلومات عقدة ما ضمن الوثيقة

هناك طرق للحركة ضمن الوثيقة لجعل عقدة ما هي العقدة الحالية بالنسبة لغرض XPathNavigator أو لإنشاء غرض XPathNavigator جديد:

الطريقة	الوصف
MoveToxxxxxx	يقوم بتحريك موقع الغرض ضمن وثيقة XML الحالية. لدينا مثلاً: MoveToFirst ,MoveToRoot ,MoveToAttribute, moveToParent,MoveToFirstChild
Clone	إنشاء غرض XPathNavigator جديد متوضع في نفس المكان الذي يتوضع فيه غرض الحالي XPathNavigator ضمن الوثيقة.

هناك أيضاً طرق مخصصة للوصول و اختيار أجزاء من محتوى وثيقة:

الطريقة	الوصف
GetAttribute	تقوم بإعادة القيمة المحددة بالمعامل المحدد من العقدة الحالية للغرض.
Select	تعيد غرض XPathNodeIterator الذي يحتوي مجموعة من العقد التي تطابق التعبير XPath.
SelectAncestors	تعيد هذه الطريقة غرض XPathNodeIterator الحاوي على مجموعة من جميع عقد السلف في الوثيقة والتي لها نمط معين واسم معين.
SelectDescendants	تعيد غرض XPathNodeIterator يحتوي مجموعة من العقد المتحدرة من الوثيقة والتي لها نمط معين أواسم معين.
SelectChildren	تعيد غرض XPathNodeItarator يحتوي مجموعة من العقد الأبناء من نمط معين أواسم معين

الغرض XmlTextWriter

- عند استخدام الغرض XmlDocument لإنشاء وثيقة XML جديدة يجب علينا إنشاء أجزاء وإدراجها ضمن الوثيقة بطريقة معينة، وهي طريقة قد تكون معقدة ومصدر للخطأ.

- يمكن للغرض XmlWriter أن يُستخدم لإنشاء وثيقة XML عقدة تلو الأخرى بطريقة تسلسلية بكتابة التأشيرات والمحتوى إلى الخرج باستخدام مجموعة الطرق التي يقدمها هذا الغرض
- يأخذ الغرض XmlTextWriter كمصدر، إما غرض TextWriter الذي يوشر إلى ملف على القرص، ومسار، واسم هذا الملف، أوغرض Stream الذي يحتوي وثيقة XML جديدة
- يقدم هذا الغرض مجموعة من الخصائص و الطرق التي يمكن استخدامها لإنشاء عقد XML و محتويات أخرى، ثم يقوم بتوجيه الخرج إلى ملف على القرص أو غرض Stream بصورة مباشرة

- يمكن أن يتم تعيين الغرض XmlTextWriter كأداة خرج لطرق في أغراض متعددة حيث يقوم غرض XmlTextWriter بتوجيه المحتوى إلى ملف على القرص أو غرض TextWriter أو غرض Stream

اكثر الطرق استخداماً لهذا الغرض هي التالية:

الطريقة	الوصف
WriteStartDocument	تقوم ببدأ و ثيقة جديدة وكتابة تصريح XML إلى الخرج.
WriteEndDocument	تقوم بإنهاء الوثيقة بإغلاق جميع العناصر غير المغلقة ثم إرسال المحتوى إلى الخرج.
WriteStartElement	تقوم بفتح تأشيرة لعنصر محدد. عندها يمكن البدء بإنشاء الواصفات باستخدام الطريقة WriteStartAttribute
WriteEndElement	تقوم بكتابة تأشيرة إغلاق للعنصر الحالي. الطريقة المقابلة لإغلاق واصفة هي WriteEndAttribute.
WriteElementString	تقوم بكتابة عنصر كامل (بما يتضمن تأشيرات الفتح و الإغلاق) باستخدام سلسة محارف كقيمة. الطريقة المقابلة لأداء نفس العلم مع الواصفات هي WriteAttributeString.
Close	تقوم بإغلاق الملف على القرص أو غرض Stream و تحرر جميع المؤشرات المرتبطة.

الغرض XmlReader

- نحتاج في بعض الأحيان إلى قراءة وثائق XML من مصدر آخر عوضاً عن كتابتها
- يعتبر الغرض XmlReader صف قاعدي يرث منه صفان عامان هما XmlTextReader و XmlNodeReader

يعتمد الغرض XmlTextReader على الغرض TextReader الذي يؤشر إلى ملف XML على القرص، وإلى مساره واسمه أوإلى غرض Stream يحتوي وثيقة XML. يمكن قراءة محتوى الوثيقة عقدة تلو الأخرى، ويوفر الغرض معلومات حول كل عقدة و حول القيمة التي تحتويها أثناء قراءتها.

يأخذ الغرض XmlNodeReader غرض XmlNode كمصدر له مما يسمح بقراءة جزء من وثيقة XML عوضاً عن قراءة كامل الوثيقة في حال كان هدفنا لوصول إلى عقدة معينة وإلى العقد الأبناء لهذه العقدة.

- يمكن أن يُستخدم الغرضان XmlTextReader و XmlNodeReader بصورة منفصلة لتوفير وصول سريع وفعال لوثائق XML أو كمصدر لأغراض أخرى حيث نقوم تلقائياً بقراءة الوثيقة وتمرير ناتج القراءة إلى الغرض الأب
- تستخدم الأغراض XPathNavigator و XmlTextReader نمذجة "pull" للوصول إلى البيانات عقدة تلو الأخرى عوضاً عن تفريغ كامل الوثيقة كشجرة ضمن الذاكرة. يسمح ذلك بالوصول إلى ملفات كبيرة الحجم دون استهلاك الكثير من الموارد ويجعل عملية كتابة النص البرمجي أسهل في أغلب الحالات.
- كما تبرز أهمية هذه الأغراض في الحالات التي نبحث فيها عن قيمة معينة عندها لن نضطر إلى قراءة كامل الوثيقة بل يمكننا الوصول إلى عقدة محددة بعد قراءة جزء من الوثيقة

يمتلك الغرضان XmlTextReader و XmlNodeReader خصائص و طرق متطابقة تقريباً، أهم الطرق المستخدمة هي:

الطريقة	الوصف
Read	تقوم بقراءة العقدة التالية إلى الغرض حيث يمكن الوصول إليها. تعيد القيمة False عندما لا يتبقى أي عقد لقراءتها.
ReadInnerXml	تقوم بقراءة وإعادة محتوى العقدة بشكل سلسلة محارف بما فيها جميع التأشيرات والنصوص للعقد الأبناء.
ReadOuterXml	تقوم بقراءة وإعادة محتوى و**تأشيرات** العقدة الحالية كسلسلة محارف بما فيها جميع التأشيرات والنصوص للعقد الأبناء .
ReadString	تقوم بإعادة سلسلة محارف تحتوي قيمة العقدة الحالية.
GetAttribute	تقوم بإعادة قيمة الواصفة المحددة للعقدة الحالية لغرض XmlReader
GetRemainder	تقوم بقراءة و إعادة الجزء الباقي من وثيقة XML المصدرية كسلسلة محارف و تعد هذه الطريقة مفيدة عندما نرغب بنقل XML من وثيقة إلى أخرى.
MoveToxxxxxx	تقوم بنقل موقع مؤشر القارئ. مثال: MoveToAttribute و MoveToContent و MoveToElement
Skip	تقوم بتجاوز العقدة الحالية والتحرك إلى العقدة التالية.
Close	تقوم بإغلاق الملف على القرص أو إغلاق الغرض Stream الذي يتم التعامل معه.

الغرض XMLValidatingReader

الغرض XMLValidatingReader:

- هناك غرض آخر مبني على الصف XmlReader وهو الغرض XmlValidatingReader. يمكننا النظر إلى هذا الغرض كغرض XmlTextReader ولكن بمهمة تقييم وثيقة مقارنةً بهيكل أو بمخطط ما
- يمكننا إنشاء غرض XmlValidatingReader من غرض XmlReadr موجود، أو من غرض Stream، أو من سلسلة محارف تحتوي XML ليتم تقييمها والتحقق منها

الغرض XslTransform:

- تعتبر الحاجة إلى تحويل الوثيقة باستخدام XSL أو XSTL من أهم متطلبات العمل مع وثائق XML
- تقدم NET. الغرض XslTransform المصمم خصيصاً لإجراء عمليات التحويل باستخدام XSL أو XSTL

تقدم XslTrasform طريقتين أساسيتين هما:

الطريقة	الوصف
Load	تقوم بتحميل وثيقة أنماط XSL و يتم التأشير إلى أي وثيقة أنماط باستخدام xsl:include
Transform	تقوم بتحويل بيانات وثيقة Xml محددةباستخدام XSL أو XSLT و تقوم بإخراج الناتج.

الغرض XMLValidatingReader :

- هناك غرض آخر مبني على الصف XmlReader و هو الغرض XmlValidatingReader.
  يمكننا النظر إلى هذا الغرض كغرض XmlTextReader و لكن بمهمة تقييم وثيقة مقارنةً بهيكل أو بمخطط ما
- يمكننا إنشاء غرض XmlValidatingReader من غرض XmlReadr موجود، أو من غرض Stream، أو من سلسلة محارف تحتوي XML ليتم تقييمها و التحقق منها

الغرض XslTransform:

- تعتبر الحاجة إلى تحويل الوثيقة باستخدام XSL أو XSTL من أهم متطلبات العمل مع وثائق XML
- تقدم NET. الغرض XslTransform المصمم خصيصاً لإجراء عمليات التحويل باستخدام XSL أو XSTL

أمثلة

نلاحظ في هذا المثال أننا استخدمنا الغرض XmlDocument ثم قمنا بتحميل ملف إليه بالطريقة Load وبعدها قمنا باستخدام الطريقة GetElementByTagname لإيجاد جميع العقد التي يكون اسم التأشيرة فيها هو "AutherName" ثم قمنا بالدوران ضمن النتائج و إظهارها.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Searching an XML document using the DOM</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Searching an XML document using the DOM</span><hr
/>
<!-----------------------------------------------------------------------
------>

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

   'create physical path to booklist.xml sample file (in same folder as
ASPX page)
   Dim strCurrentPath As String = Request.PhysicalPath
   Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

   'create a new XMLDocument object
   Dim objXMLDoc As New XmlDocument()

   Try

      'load the XML file into the XMLDocument object
      objXMLDoc.Load(strXMLPath)
      outDocURL.innerHTML = "Loaded file: <b>" & strXMLPath & "</b>"

   Catch objError As Exception

      'display error details
```

```
      outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
          & objError.Message & "<br />" & objError.Source
      Exit Sub   ' and stop execution

   End Try

   'now ready to parse the XML document
   'it must be well-formed to have loaded without error

   'create a string to hold the matching values found
   Dim strResults As String = "<b>List of authors</b>:<br />"

   'create a NodeList collection of all matching child nodes
   Dim colElements As XmlNodeList
   colElements = objXMLDoc.GetElementsByTagname("AuthorName")

   'iterate through the collection getting the values of the
   'child #text nodes for each one
   Dim objNode As XmlNode
   For Each objNode In colElements
      strResults += objNode.FirstChild().Value & "<br />"
   Next

   'then display the result
   outResults.innerHTML = strResults   'display the result

End Sub

</script>

</body>
</html>
```

في المثال التالي سنقوم باستخدام غرض XmlDocument ثم تحميل وثيقة XML إليه بواسطة الطريقة Load.

قمنا بتعريف تابع ليقوم بإعادة نوع العقدة ضمن الوثيقة بناءاً على الرقم الذي تعيده الطريقة NodeType.

ثم قمنا بتعريف تابع آخر يقوم باستعراض العقد وفحص العقد الأبناء لكل عقدة.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Accessing XML documents using the DOM</title>
<!-- #include file="..\global\style.inc" -->
```

```
</head>
<body bgcolor="#ffffff">
<span class="heading">Accessing XML documents using the DOM</span><hr
/>
<!--------------------------------------------------------------------
------>

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

   'create physical path to booklist.xml sample file (in same folder as
ASPX page)
   Dim strCurrentPath As String = Request.PhysicalPath
   Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

   'create a new XMLDocument object
   Dim objXMLDoc As New XMLDocument()

   Try

      'load the XML file into the XMLDocument object
      objXMLDoc.Load(strXMLPath)
      outDocURL.innerHTML = "Loaded file: <b>" & strXMLPath & "</b>"

   Catch objError As Exception

      'display error details
      outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
          & objError.Message & "<br />" & objError.Source
      Exit Sub  ' and stop execution

   End Try

   'now ready to parse the XML document
   'it must be well-formed to have loaded without error
   'call a recursive function to iterate through all the nodes
   'in the document creating a string that is placed in the <div> above
   Dim strNodes As String
   outResults.innerHTML = strNodes &
GetChildNodes(objXMLDoc.ChildNodes, 0)
```

```
End Sub



Function GetChildNodes(objNodeList As XMLNodeList, intLevel As Integer)
As String

  Dim strNodes As String = ""
  Dim objNode As XMLNode
  Dim objAttr As XMLAttribute

  'iterate through all the child nodes for the current node
  For Each objNode In objNodeList

     'display information about this node
     strNodes = strNodes & GetIndent(intLevel) _
              & GetNodeType(objNode.NodeType) & ": <b>" & objNode.Name

     'if it is an XML Declaration node, display the 'special'
properties
     If objNode.NodeType = XMLNodeType.XmlDeclaration Then
        'cast the XMLNode object to an XmlDeclaration object
        Dim objXMLDec =CType(objNode, XmlDeclaration)
        strNodes = strNodes & "</b>  version=<b>" &
objXMLDec.Version _
                 & "</b>  standalone=<b>" & objXMLDec.Standalone &
"</b><br />"
     Else
        'just display the generic 'value' property
        strNodes = strNodes & "</b>  value=<b>" & objNode.Value &
"</b><br />"
     End If

     'if it is an Element node, iterate through the Attributes
     'collection displaying information about each attribute
     If objNode.NodeType = XMLNodeType.Element Then
        'display the attribute information for each attribute
        For Each objAttr In objNode.Attributes
           strNodes = strNodes & GetIndent(intLevel + 1) _
                    & GetNodeType(objAttr.NodeType) & ": <b>" &
objAttr.Name _
                    & "</b>  value=<b>" & objAttr.Value & "</b><br
/>"
        Next
     End If

     'if this node has child nodes, call the same function recursively
     'to display the information for it and each of its child node
```

```
      If objNode.HasChildNodes Then
       strNodes = strNodes & GetChildNodes(objNode.childNodes, intLevel
+ 1)
      End If

   Next  'go to next node

   Return strNodes   'pass the result back to the caller

End Function



Function GetIndent(intLevel As Integer)
   'returns a string of non-breaking spaces used to indent each line
   Dim strIndent As String = ""
   Dim intIndent As Integer
   For intIndent = 0 To intLevel
     strIndent = strIndent & "      "
   Next
   Return strIndent
End Function

Function GetNodeType(intType As Integer) As String
    'returns the node type as a string
    Select Case (intType)
      Case 0: Return "NONE"
      Case 1: Return "ELEMENT"
      Case 2: Return "ATTRIBUTE"
      Case 3: Return "TEXT"
      Case 4: Return "CDATA SECTION"
      Case 5: Return "ENTITY REFERENCE"
      Case 6: Return "ENTITY"
      Case 7: Return "PROCESSING INSTRUCTION"
      Case 8: Return "COMMENT"
      Case 9: Return "DOCUMENT"
      Case 10: Return "DOCUMENT TYPE"
      Case 11: Return "DOCUMENT FRAGMENT"
      Case 12: Return "NOTATION"
      Case 13: Return "WHITESPACE"
      Case 14: Return "SIGNIFICANT WHITESPACE"
      Case 15: Return "END ELEMENT"
      Case 16: Return "END ENTITY"
      Case 17: Return "XML DECLARATION"
      Case 18: Return "NODE (ALL)"
      Case Else: Return "UNKNOWN"
    End Select
End Function

</script>
```

```
</body>
</html>
```

سنستخدم في المثال التالي الغرض XmlDocument مع الغرض XPathNavigator لن نستخدم هنا الغرض XMLNode وسنعيد نفس المثال السابق.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.Xml" %>
<%@Import Namespace="System.Xml.Xpath" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Accessing XML documents using an XPathNavigator</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Accessing XML documents using an
XPathNavigator</span><hr />
<!------------------------------------------------------------------------
------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

   'create physical path to booklist.xml sample file (in same folder as
ASPX page)
   Dim strCurrentPath As String = Request.PhysicalPath
   Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

   'create a new XmlDocument object
   Dim objXMLDoc As New XmlDocument

   Try

      'load the XML file
      objXMLDoc.Load(strXMLPath)
      outDocURL.innerHTML = "Loaded file: <b>" & strXMLPath & "</b>"

   Catch objError As Exception
```

```
      'display error details
      outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
          & objError.Message & "<br />" & objError.Source
      Exit Sub  ' and stop execution

   End Try

   'now ready to parse the XML document
   'it must be well-formed to have loaded without error
   'create a new XPathNavigator object using the XMLDocument object
   Dim objXPNav As XPathNavigator = objXMLDoc.CreateNavigator()

   'move the current position to the root #document node
   objXPNav.MoveToRoot()

   'call a recursive function to iterate through all the nodes in the
   'XPathNavigator, creating a string that is placed in the <div> above
   outResults.innerHTML = GetXMLDocFragment(objXPNav, 0)

End Sub



Function GetXMLDocFragment(objXPNav As XPathNavigator, intLevel As
Integer) As String

  Dim strNodes As String = ""
  Dim intLoop As Integer

  'display information about this node
  strNodes = strNodes & GetIndent(intLevel) _
           & GetNodeType(objXPNav.NodeType) & ": <b>" & objXPNav.Name _
           & "</b>  value=<b>" & objXPNav.Value & "</b><br />"

  'see if this node has any Attributes
  If objXPNav.HasAttributes Then

     'move to the first attribute
     objXPNav.MoveToFirstAttribute()

     Do

        'display the information about it
        strNodes = strNodes & GetIndent(intLevel + 1) _
                 & GetNodeType(objXPNav.NodeType) & ": <b>" &
objXPNav.Name _
```

```
                & "</b>  value=<b>" & objXPNav.Value & "</b><br
/>"

     Loop While objXPNav.MoveToNextAttribute()

     'then move back to the parent node (i.e. the element itself)
     objXPNav.MoveToParent()

  End If

  'see if this node has any child nodes
  If objXPNav.HasChildren Then

     'move to the first child node of the current node
     objXPNav.MoveToFirstChild()

     Do
       'recursively call this function to display the child node
fragment
       strNodes = strNodes & GetXMLDocFragment(objXPNav, intLevel + 1)
     Loop While objXPNav.MoveToNext()

     'move back to the parent node - the node we started from when we
     'moved to the first child node - could have used Push and Pop
instead
     objXPNav.MoveToParent()

  End If

  'must repeat the process for the remaining sibling nodes (i.e.nodes
  'at the same 'level' as the current node within the XML document
  'so repeat while we can move to the next sibling node
  Do While objXPNav.MoveToNext()

     'recursively call this function to display this sibling node
     'and its atributes and child nodes
     strNodes = strNodes & GetXMLDocFragment(objXPNav, intLevel)

  Loop

  Return strNodes  'pass the result back to the caller

End Function
```

```
Function GetIndent(intLevel As Integer)
  'returns a string of non-breaking spaces used to indent each line
  Dim strIndent As String = ""
  Dim intIndent As Integer
  For intIndent = 0 To intLevel
    strIndent = strIndent & "      "
  Next
  Return strIndent
End Function


Function GetNodeType(intType As Integer) As String
   'returns the node type as a string
   Select Case (intType)
     Case 0: Return "NONE"
     Case 1: Return "ELEMENT"
     Case 2: Return "ATTRIBUTE"
     Case 3: Return "TEXT"
     Case 4: Return "CDATA SECTION"
     Case 5: Return "ENTITY REFERENCE"
     Case 6: Return "ENTITY"
     Case 7: Return "PROCESSING INSTRUCTION"
     Case 8: Return "COMMENT"
     Case 9: Return "DOCUMENT"
     Case 10: Return "DOCUMENT TYPE"
     Case 11: Return "DOCUMENT FRAGMENT"
     Case 12: Return "NOTATION"
     Case 13: Return "WHITESPACE"
     Case 14: Return "SIGNIFICANT WHITESPACE"
     Case 15: Return "END ELEMENT"
     Case 16: Return "END ENTITY"
     Case 17: Return "XML DECLARATION"
     Case 18: Return "NODE (ALL)"
     Case Else: Return "UNKNOWN"
   End Select
End Function

</script>

<!-----------------------------------------------------------------------
------>
</body>
</html>
```

سنقوم بعمل مشابه للأمثلة السابقة هذه المرة باستخدام الغرض XmlTextReader.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<html><head>
<title>Accessing an XML document with an XMLTextReader object</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Accessing an XML document with an XMLTextReader
object</span><hr />
<!----------------------------------------------------------------------
------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

   'create physical path to booklist.xml sample file (in same folder as
ASPX page)
   Dim strCurrentPath As String = Request.PhysicalPath
   Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

   'declare a variable to hold an XmlTextReader object
   Dim objXMLReader As XmlTextReader

   Try

      'create a new XmlTextReader object for the XML file
      objXMLReader = New XmlTextReader(strXMLPath)
      outDocURL.innerHTML = "Opened file: <b>" & strXMLPath & "</b>"

   Catch objError As Exception

      'display error details
      outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
          & objError.Message & "<br />" & objError.Source
      Exit Sub  ' and stop execution

   End Try

   'now ready to read (or "pull") the nodes of the XML document
   Dim strNodeResult As String = ""
   Dim objNodeType As XmlNodeType
```

```
   'read each node in turn - returns False if no more nodes to read
   Do While objXMLReader.Read()

      'select on the type of the node (these are only some of the
types)
      objNodeType = objXMLReader.NodeType

      Select Case objNodeType

        Case XmlNodeType.XmlDeclaration:
           'get the name and value
           strNodeResult += "XML Declaration: <b>" & objXMLReader.Name
_
                         & " " & objXMLReader.Value & "</b><br />"

        Case XmlNodeType.Element:
           'just get the name, any value will be in next (#text) node
           strNodeResult += "Element: <b>" & objXMLReader.Name &
"</b><br />"

        Case XmlNodeType.Text:
           'just display the value, node name is "#text" in this case
           strNodeResult += "  - Value: <b>" & objXMLReader.Value
_
                         & "</b><br />"

      End Select

      'see if this node has any attributes
      If objXMLReader.AttributeCount > 0 Then

         'iterate through the attributes by moving to the next one
         'could use MoveToFirstAttribute but MoveToNextAttribute does
         'the same when the current node is an element-type node
         Do While objXMLReader.MoveToNextAttribute()

            'get the attribute name and value
            strNodeResult +=  "  - Attribute: <b>" &
objXMLReader.Name _
                          & "</b>   Value: <b>" &
objXMLReader.Value _
                          & "</b><br />"
         Loop

      End If

   Loop   'and read the next node
```

```
    'finished with the reader so close it
    objXMLReader.Close()

    'and display the results in the page
    outResults.innerHTML = strNodeResult

End Sub

</script>
<!--------------------------------------------------------------------
------>
</body>
</html>
```

في المثال التالي سنقوم باستخدام الغرض XMLTextWriter و سنقوم بكتابة وثيقة XMl بواسطته. نلاحظ أننا بدأنا بإنشاء الغرض XmlTextWriter ثم قمنا بتحديد التنسيق (مقدار الإزاحة بين العناصر) من ثم بدأنا بكتابة الوثيقة باستخدام طرق الغرض مثل WriteStartDocument و WriteComment و WriteStartElement....

ثم قمنا بتفريغ محتوى الغرض إلى الملف و أغلقنا الملف باستخدام الطريقتين flush و Close.

بعدها قمنا بفتح الملف و تحميل محتواه إلى سلسة محارف ثم قمنا بكتابة المحتوى إلى عنصر التحكم من جهة المخدم OurResult.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>
<%@ Import Namespace="System.IO" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Creating an XML document with an XMLTextWriter object</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Creating an XML document with an XMLTextWriter
object</span><hr />
<!--------------------------------------------------------------------
------>

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()
```

```
   'create physical path for the new file (in same folder as ASPX page)
   Dim strCurrentPath As String = Request.PhysicalPath
   Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "newbooklist.xml"

   'declare a variable to hold an XmlTextWriter object
   Dim objXMLWriter As XmlTextWriter

   Try

      'create a new objXMLWriter object for the XML file
      objXMLWriter = New XmlTextWriter(strXMLPath, Nothing)
      outDocURL.innerHTML = "Writing to file: <b>" & strXMLPath &
"</b>"



   Catch objError As Exception

      'display error details
      outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
          & objError.Message & "<br />" & objError.Source
      Exit Sub  ' and stop execution

   End Try

   'now ready to write (or "push") the nodes for the new XML document

   'turn on indented formatting and set indent to 3 chararcters
   objXMLWriter.Formatting = Formatting.Indented
   objXMLWriter.Indentation = 3

   'start the document with the XML declaration tag
   objXMLWriter.WriteStartDocument()

   'write a comment element including the current date/time
   objXMLWriter.WriteComment("Created using an XMLTextWriter - " &
Now())

   'write the opening tag for the <BookList> root element
   objXMLWriter.WriteStartElement("BookList")

      'write the opening tag for a <Book> element
      objXMLWriter.WriteStartElement("Book")

         'add two attributes to this element's opening tag
```

```
        objXMLWriter.WriteAttributeString("Category", "Technology")
        Dim intPageCount As Integer = 1248   'numeric value to convert
        objXMLWriter.WriteAttributeString("Pagecount",
intPageCount.ToString("G"))

        'write four elements, using different source data types
        objXMLWriter.WriteElementString("Title", "Professional Video
Recorder Programming")
        Dim datReleaseDate As DateTime = #03/03/2000#
        objXMLWriter.WriteElementString("ReleaseDate",
datReleaseDate.ToString("yyyy-MM-dd"))
        Dim intSales As Integer = 17492
        objXMLWriter.WriteElementString("Sales",
intSales.ToString("G"))
        Dim blnHardback As Boolean = True
        objXMLWriter.WriteElementString("Hardback",
blnHardback.ToString())

        'write the opening tag for the <AuthorList> child element
        objXMLWriter.WriteStartElement("AuthorList")

           'add two <Author> elements
           objXMLWriter.WriteElementString("Author", "Francesca Unix")
           objXMLWriter.WriteElementString("Author", "William Soft")

        'close the <AuthorList> element
        objXMLWriter.WriteEndElement()

      'close the <Book> element
      objXMLWriter.WriteEndElement()

   'close the root <BookList> element
   objXMLWriter.WriteEndElement()

   'flush the current content to the file and close it
   objXMLWriter.Flush()
   objXMLWriter.Close()

   'now open the new XML file and read it into a string
   Dim strXMLResult As String
   Dim objSR As StreamReader = File.OpenText(strXMLPath)
   strXMLResult = objSR.ReadToEnd()
   objSR.Close
   objSR = Nothing

   'and display the results in the page
   outResults.innerHTML = "<pre>" & Server.HtmlEncode(strXMLResult) &
"<pre>"
```

```
End Sub
</script>
<!-----------------------------------------------------------------------
------>


</body>
</html>
```

الفصل الثاني عشر

استخدام AJAX

يتعرف الدارس في هذا الفصل على:

- لماذا AJAX
- حسنات و سيئات AJAX
- أدوات تطوير AJAX
- التحديث الجزئي في AJAX

AJAX

تعلمنا في الجلسات الماضية كيفية بناء صفحات وب تستخدم نموذج postback.
يتم في هذا النموذج إعادة إرسال الصفحات بشكل مستمر إلى المخدم لتتم إعادة توليدها.
فإذا أخذنا مثلاً مثال بطاقة التهنئة الذي عملنا عليه في جلسة سابقة، ستتم إعادة إرسال الصفحة في كل مرة يختار فيها المستخدم نوع خط أو لون أو يدخل نصاً ما.
يستدعي هذا من ASP.NET إعادة تصيير الصفحات مراراً وتكراراً وإعادة استلام المتصفح لنسخة جديدة من الصفحة.
يتطلب هذا السيناريو عملاً كثيراً وإن كان الزمن اللازم لإعادة توليد الصفحة مقبولاً نسبياً لكن هناك وقت ضائع في إعادة توليد وإرسال واستقبال الأجزاء التي لم تتغير من الصفحة.
كما تعد عملية إعادة الإرسال مشتتة بعض الشيء للمستخدم الذي قد تتم مقاطعته أثناء إدخاله للمعلومات ضمن عنصر تحكم آخر أو زلق الصفحة إلى بدايتها.
لذلك ستفتقر تجربة استخدام التطبيق الناتجة إلى التجاوب الذي تمنحه تطبيقات سطح المكتب.
ظهر مؤخراً جيل جديد من تطبيقات الوب التي تؤمن للمستخدم تفاعلاً شبيهاً بذلك الذي يمكن الحصول عليه في تطبيقات سطح المكتب.
تحدّث هذه التطبيقات نفسها بشكل سريع وتقدم ميزات خاصة مثل الحركة وقابلية الجر والإلقاء.
من الأمثلة الشائعة على هذه التطبيقات البريد الالكتروني الذي توفره Google والأدوات مثل Google Maps.
يستخدم هذا الجيل الجديد من تطبيقات الوب ممارسات في التصميم وتقنيات تعرف باسم AJAX.
AJAX اختصار لمجموعة من التقنيات التي تساعد على إنشاء صفحات ديناميكية أكثر تجاوباً.
تعتبر القدرة على تحديث جزء من الصفحة دون التأثير على الأجزاء الأخرى أحد الدعامات الأساسية في AJAX.

كيف نفهم AJAX – الحسنات

لتشكيل فهم واضح لتقنية AJAX لا بد من البدء بتحديد حسنات وسيئات هذه المقاربة:
الحسنات:

- تعتبر الفائدة الأساسية لتقنية AJAX هي الاستجابة.

نحصل عند تصميم تطبيق AJAX بشكل صحيح على تجربة أفضل من وجهة نظر المستخدم.

وإن كان المستخدم لا يستطيع أن يحصل على أي جديد أو على سرعة أكبر لكن تظهر تجربة الاستخدام بشكل أكثر نضوجاً وتكلفاً وبالأخص إذا كان التطبيق الذي نقوم بتطويره ينافس تطبيقات أخرى مماثلة لا تستخدم هذه التقنية.

- توفر AJAX إضافة إلى هذا ميزات غير ممكنة في صفحات الوب التقليدية. على سبيل المثال تستخدم AJAX بشكل كبير رماز JavaScript الذي سيستجيب للأحداث من طرف الزبون مثل حركة مؤشر الفأرة وضغط الأزرار.

حيث تعتبر هذه الأحداث عالية التكرارية مما يجعل من غير العملي التعامل معها باستخدام نموذج postback.

مثلاً في حال أردنا تمييز علبة نصية عندما يتحرك مؤشر الفأرة فوقها سنحتاج في حال اخترنا نموذج postback إلى إعادة إرسال كامل الصفحة إلى مخدم الوب، إعادة توليدها وتحديث الصفحة ضمن المستعرض في الوقت الذي قد يكون فيه مؤشر الفأرة قد تحرك إلى مكان ثانٍ كلياً.

تستطيع AJAX الاستجابة مع مثل هذا السيناريو والتجاوب بشكل آني، تحديث الصفحة عند الحاجة وطلب أي معلومات من مخدم الوب في الخلفية عند الحاجة إليها.

يملك المستخدم الحرية في استمرار التعامل مع الصفحة خلال فترة إرسال الطلب إلى المخدم حتى أنه لن يشعر بإرسال الطلب أصلاً.

ملاحظة: ليست AJAX في الحقيقة تقنية جديدة ولكنها مجموعة من الممارسات التي تسمح باستثمار JavaScript وغرض XMLHttpRequest الذي توفره بعض المستعرضات.

كيف نفهم AJAX – السيئات

السيئات:

يشمل استخدام AJAX تحديين أساسيين الأول مرتبط بالتعقيد حيث تعتبر عملية كتابة نص JavaScript البرمجي في تطبيق AJAX معقدة.

لحسن الحظ ستقوم ASP.NET بإدارة هذا الموضوع عوضاً عن المطور إذ توفر مجموعة من الميزات التي تسهل التعامل مع AJAX.

يكمن التحدي الثاني في مدى دعم المستعرض لـــAJAX.

ظهرت التقنية التي تدعم AJAX منذ عدة أعوام ولكنه لم يتم دعمها بشكل كامل على معظم المستعرضات حتى وقت قريب.

إذا قمنا بالتفاعل مع تطبيق AJAX باستخدام مستعرض IE يجب التأكد من كون النسخة أحدث من النسخة 5 كذلك هي الحال بالنسبة للمستعرضات الأخرى كما يوضح الجدول التالي:

المستعرض	النسخة
Internet Explorer	5
Netscape	7
Opera	7.6
Safari	1.2
FireFox	1.0

يتعلق قرار استخدام AJAX بمستخدمي التطبيق لكن يمكن الجزم بأن AJAX ستعمل بشكل طبيعي على 90% من مستعرضات الزبائن.

ولكن هل معنى هذا أن مستخدمي المستعرضات القديمة لن يتمكنوا من استخدام التطبيق؟

ترتبط الإجابة بالميزة المستخدمة ففي حال استخدام ميزة التصيير الجزئي التي توفرها ASP.NET من خلال عنصر تحكم UpdatePanel مثلاً ستعمل الصفحة بشكل طبيعي حتى على المستعرضات التي لا تدعم AJAX حيث سيتم استخدام إعادة الإرسال الكلية عوضاً عن التحديث الجزئي.

أما في حال استخدام عناصر تحكم AJAX أكثر تعقيداً سنجد أنها لن تعمل بشكل صحيح على المستعرضات القديمة.

يكفي للتأكد من دعم صفحات التطبيق للمستعرضات القديمة إلغاء تفعيل JavaScript من المستعرض وإعادة تجريب الصفحات.

لا بد أن نعلم هنا أن هناك ثمن سندفعه لقاء استخدام ميزات AJAX ويرتفع هذا الثمن كلما تطلبت الميزة دعماً قد لا توفره معظم المستعرضات.

أخيراً تقوم AJAX بالكثير من الأعمال ضمن صفحة واحدة وهذا المفهوم قد لا يكون مرغوباً ولا ينسجم مع مايتم عادة ضمن صفحات الوب التقليدية.

ملاحظة: قد تؤثر هذه المقاربة على طريقة المطور في تصميم التطبيق بشكل عام.

أدوات تطوير AJAX في ASP.NET

هناك عدة طرق لاستخدام AJAX في أي تطبيق بما فيها تطبيقات ASP.NET.

لتطوير تطبيقات AJAX دون الحاجة إلى الأدوات التي توفرها ASP.NET يكفي أن يمتلك المطور فهماً كافياً لـــJavaScript كونها اللغة الأساسية التي يستخدمها المستعرض ويطلب بواسطتها المعلومات من مخدم الوب حسب الحاجة.

بالرغم من كون JavaScript غير شديدة التعقيد لكنه يصعب كتابة برنامج صحيح بواسطتها وذلك لسببين:

- تختلف تفاصيل تطبيق JavaScript من مستعرض إلى آخر مما يتطلب خبرة كبيرة لدى المطور لكتابة صفحة مستقرة قادرة على العمل بشكل صحيح على كل المستعرضات.
- يؤخذ على JavaScript أنها لغة سيئة السمعة تتغاضى عن الكثير من الأخطاء مما يجعل عملية تحديد مكان هذه الأخطاء وإزالتها عملية صعبة حتى أن بعض هذه الأخطاء قد يكون قاتلاً في بعض المستعرضات ويتم تجاوزه دون أي أثر في مستعرضات أخرى مما يعقد عملية إزالة العلل من النص البرمجي.

سنركز في هذه الجلسة على استخدام نموذج بسوية أعلى لتطوير تطبيقات AJAX وذلك بالاعتماد على ما تقدمه ASP.NET من مكونات وعناصر تحكم من طرف المخدم خاصة بهذا الغرض.

تقوم عناصر التحكم آلياً بتصيير نص JavaScript المطلوب للحصول على التأثيرات المرغوبة.

يمكن بهذه الطريقة بناء تطبيقات AJAX مع استمرار المحافظة على العمل في بيئة مألوفة وبإنتاجية عالية.

لن نحصل باستخدام أدوات AJAX ضمن ASP.NET على مرونة وقدرة كبيرة على التخصيص ولكننا سنكون قادرين على الحصول على تطبيقات عاملة وبأقل جهد ممكن.

مدير النص

لنتمكن من استخدام AJAX ضمن ASP.NET نحتاج إلى استعمال عنصر تحكم وب جديد هو ScriptManager حيث يعد عنصر التحكم هذا لب ASP.NET AJAX.

يتوضع عنصر التحكم هذا كباقي عناصر التحكم على شريط الأدوات ضمن التبويب AJAX ويؤدي سحبه إلى الصفحة إلى إضافة

التأشيرة التالية:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
```

يظهر عنصر تحكم ScriptManager في زمن التصميم كعلبة فارغة فضية اللون ولكن عند استدعاء صفحة تحتوي على عنصر التحكم هذا لن نرى أي شيء مكانه.

لا يقوم عنصر التحكم هذا بتوليد أي تأشيرات HTML ولكن يقوم بإضافة روابط إلى مكتبات JavaScript الخاصة بـــAJAX وذلك بإضافة نصوص خطاطية شبيهة بالشكل التالي:

```
<script src="/YourWebSite/ScriptResource.axd?d=RUSU1ml ..."
type="text/javascript">
</script>
```

نلاحظ هنا أن النص الخطاطي استخدم الواصفة src للوصول إلى نص JavaScript ولكن تم استخدام صيغة مسار لاتعبر في الحقيقة عن اسم ملف لكنها عنوان يخبر ASP.NET بإيجاد الملف المتضمن في ملفات NET 3.5. المترجمة إلى لغة التجميع.

تحدد سلسلة الاستعلام المحرفية في نهاية عنوان URL للاحقة ScriptResource.axd ما هو الملف الذي يجب إرساله إلى المستعرض.

ملاحظة: لا تتطلب الصفحات الحاوية على AJAX زمناً أطول نسبياً لتحميلها من المخدم كون الملفات المستخدمة مضغوطة، يتم حفظها ضمن ذاكرة الخبئ في المستعرض وأحياناً يتم نقلها بشكل مضغوط من المخدم مباشرة في حال دعم المستعرض لهذه الميزة.

تتطلب كل صفحةِ AJAX استخدام مثيل لغرض ScriptManager.

تقوم عناصر التحكم الخاصة بـــ Ajax بالتفاعل مع غرض ScriptManager لتطلب منه تأسيس ارتباطات إلى موارد Javascript إضافية.

ملاحظة في حال الحاجة إلى استخدام ميزات AJAX في معظم صفحات الموقع يمكن أن نقوم بوضع عنصر تحكم ScriptManager ضمن الصفحة الأساسية (master page).

قد يتسبب هذا السيناريو بمشكلة نابعة من ضرورة تغيير إعدادات خصائص غرض ScriptManager لتخصيصها بكل صفحة.

يكون الحل في هذه الحالة استخدام غرض ScriptManagerProxy في كل صفحة مما يسمح بتحديد إعدادات مختلفة.

مواضيع هامة

سنقوم ضمن ما تبقى من هذه الجلسة بتغطية مجموعة من المواضيع المرتبطة ببناء صفحات AJAX وهي:

- استخدام التحديث الجزئي لتجنب إعادة إرسال الصفحة.
- استخدام مؤشر التقدم للتعامل مع التحديثات البطيئة.
- استخدام التحديث وفق جدول زمني لتحديث جزء من الصفحة.

استخدام عناصر تحكم أدوات تطوير AJAX في ASP.NET.

التحديث الجزئي

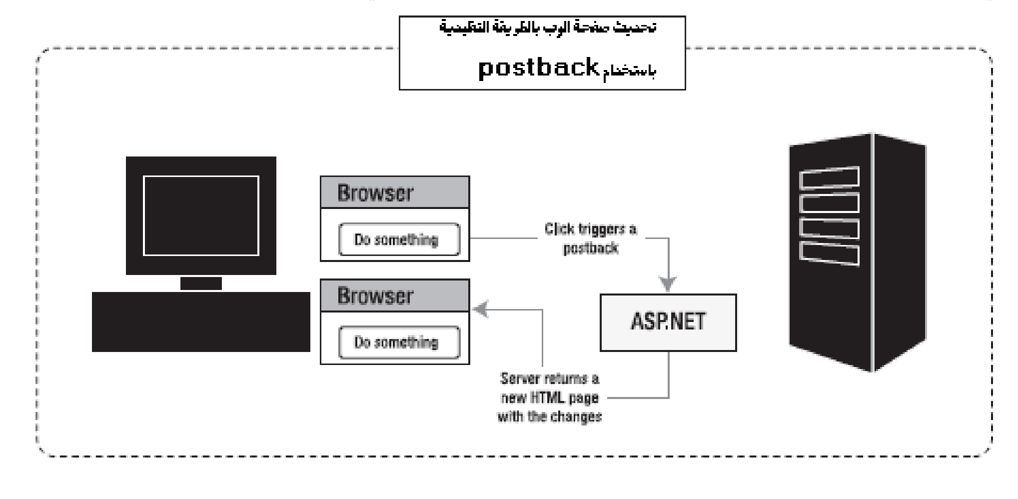
يعتبر التحديث الجزئي تقنية أساسية في تطبيقات AJAX.

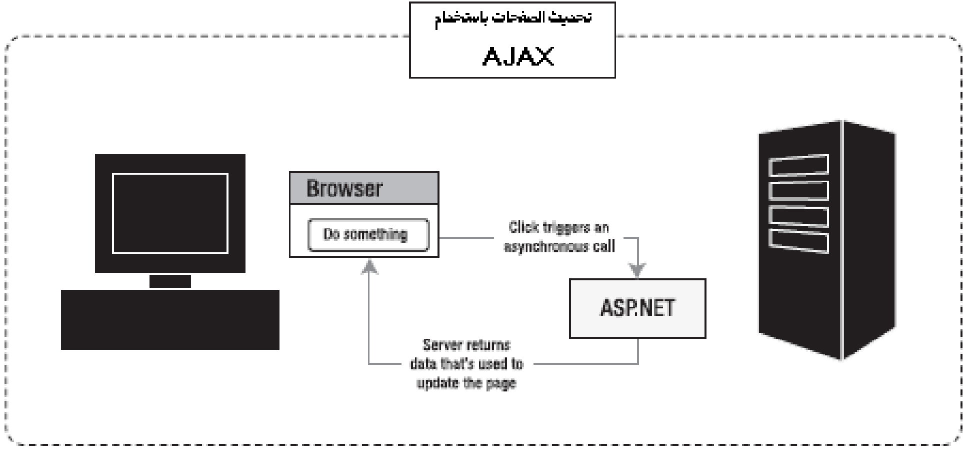
في هذه المقاربة لا نحتاج إلى إعادة إرسال الصفحة وتحديثها من قبل المستعرض ولكن عند حصول شيء ما تقوم الصفحة بطلب

المزيد من المعلومات من المخدم.

تتم هذه العملية في الخلفية مما يحافظ على استجابة صفحة الوب.

يوضح الشكل التالي الفرق بين استخدام AJAX وتحديث الصفحات بشكل تقليدي في ASP.NET:





تتضمن ASP.NET عناصر تحكم مفيدة تسمح بتحويل صفحة عادية تستخدم منطقاً من جهة المخدم وتحديثها بدون رجفة في الصفحة باستخدام التحديث الجزئي.

عنصر التحكم هذا هو UpdatePanel.

الفكرة الأساسية هي تقسيم صفحة الوب إلى منطقة مستقلة أو أكثر وتغليف كل منها بعنصر تحكم UpdatePanel غير المرئي.

عند انطلاق حدث في أحد عناصر التحكم المحتواة في عنصر تحكم UpdatePanel سيتم اعتراض عملية إرسال الصفحة واستبدالها باستدعاء غير متزامن.

يتم العمل وفق المراحل التالية:

1- يقوم المستخدم بنقر الزر ضمن عنصر تحكم UpdatePanel.

2- يقوم عنصر التحكم UpdatePanel باعتراض حدث النقر واستبداله بطلب غير متزامن.

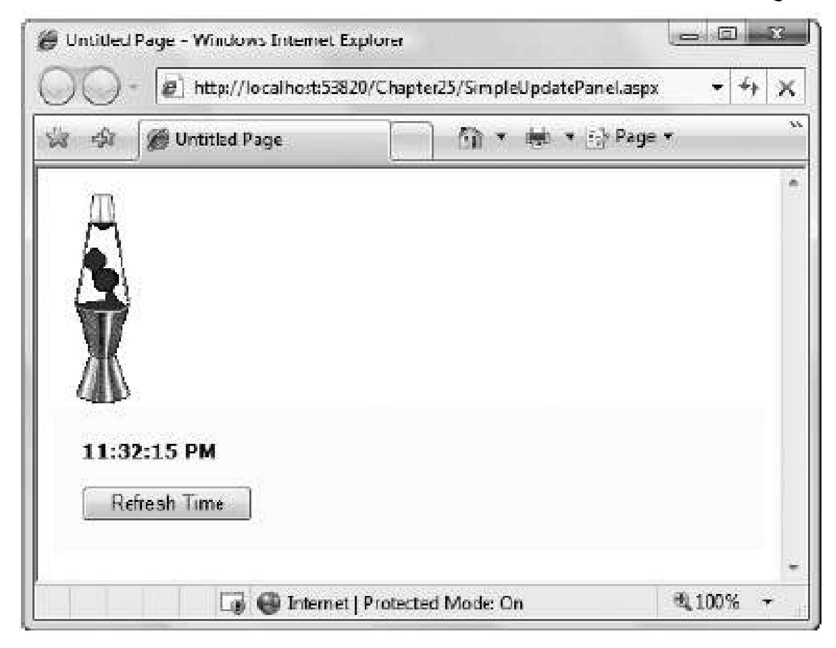
3- يتم تنفيذ دورة حياة الصفحة الطبيعية من طرف المخدم وتصيير الناتج كنص HTML وإعادته إلى المستعرض.

4- تستقبل ASP.NET AJAX كامل صفحة HTML وتقوم باستبدال محتوى عنصر التحكم بالمحتوى الجديد.

لن يتم تعديل أي شيء غير محتوى ضمن عنصر تحكم UpdatePanel.

مثال على استخدام UpdatePanel

يستخدم المثال التالي عنصر تحكم UpdatePanel لتحديث الزمن الحالي دون إعادة إرسال كامل الصفحة أو التأثير على المكونات الأخرى.



نلاحظ أنه لا بد من استخدام عنصر تحكم ScriptManager قبل عنصر تحكم UpdatePanel لأن الصفحة تحتاج إلى الحصول على نصوص Javascript قبل أن تستطيع استخدامها.

```
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

<img src="myAnimatedImage.gif">

<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
<ContentTemplate>
<div style="background-color:LightYellow;padding: 20px">
<asp:Label ID="lblTime" runat="server" Font-Bold="True"></asp:Label>
<br />
<br />
<asp:Button ID="cmdRefreshTime" runat="server"
OnClick="cmdRefreshTime_Click"
Text="Refresh Time" />
</div>
</ContentTemplate>
</asp:UpdatePanel>
```

أما النص البرمجي الخاص بنقر زر التحديث فهو:

```
protected void cmdRefreshTime_Click(object sender, EventArgs e)
{
lblTime.Text = DateTime.Now.ToLongTimeString();
}
```

الثمن الذي سندفعه لقاء هذه الميزة هو بعض الوقت الإضافي اللازم للعملية.

أحد نقاط الضعف في هذه المقاربة والتي تجعلها مختلفة عن عملية بناء نص JavaScript يدوياً هي الاضطرار إلى تحميل كامل الصفحة وإن كان القسم الذي سيتم الاستفادة منه يقتصر على الزمن والتاريخ الحالي.

معالجة الأخطاء

ذكرنا سابقاً أن عنصر التحكم UpdatePanel لا يؤثر على الآلية التي يتم فيها العمل على المخدم وأن الفارق يقتصر على آلية الاتصال حيث تتم العملية باستخدام استدعاء غير متزامن.

عند حدوث استثناء غير معالج يتم التقاط الخطأ من قبل غرض ScriptManager وتمريره إلى الزبون.

يقوم نص JavaScript من جهة الزبون بتوليد خطأ JavaScript.

ما يحدث لاحقاً منوط بالمستعرض المستخدم ولكن عادة ما تتجاوز معظم المستعرضات الأخطاء بشكل صامت. تظهر في IE مثلاً رسالة ضمن شريط الحالة تحدد المشكلة.

هناك خيارات أخرى للتعامل مع الأخطاء التي تظهر عند استخدام استدعاء غير متزامن من هذه الطرق استخدام صفحات الأخطاء المخصصة.

كل ما نحتاجه للتعامل مع هذه الصفحات إضافة العنصر <customErrors> إلى الملف web.config بصيغة مشابهة لما يلي:

```
<customErrors defaultRedirect="ErrorPage.aspx" mode="On"></customErrors>
```

عند تلقي PageRequestManager خطأ سيقوم بإعادة توجيه المستعرض إلى الصفحة ErrorPage.aspx وإضافة معامل aspxerrorpath إلى السلسلة المحرفية للاستعلام ضمن محدد URL للصفحة التي سببت المشكلة.

```
http://localhost/Ajax/ErrorPage.aspx?aspxerrorpath=/Ajax/UpdatePanels.aspx
```

يمكن كتابة نص برمجي ضمن الصفحة ErrorPage.aspx يقوم بقراءة معلومات المعامل aspxerrorpath كما في المثال التالي:

```
string url = Request.QueryString["aspxerrorpath"];
if (url != null) Response.Redirect(url);
```

ملاحظة: عند استخدام صفحات الأخطاء المخصصة لا بد من إسناد القيمة false إلى الخاصة ScriptManagerAllowCustomErrorsRedirect.

ملاحظة: تتضمن ASP.NET3.5 عنصري تحكم لا يمكن استخدامها مع UpdatePanel وهي عنصر تحكم FileInput و HtmlInputFile.

التحديث الشرطي

قد تحتوي صفحات الوب المعقدة على أكثر من عنصر تحكم UpdatePanel.

في هذه الحالة عند إطلاق أحد هذه العناصر لعملية التحديث سيتم تحديث جميع مناطق عناصر التحكم تلك.

لتلافي عملية تحديث جميع عناصر تحكم UpdatePanel نلجأ إلى إسناد القيمة Conditional إلى الخاصة UpdateMode لعناصر التحكم عوضاً عن القيمة Always.

ستقوم عناصر UpdatePanel بتحديث نفسها فقط إذا تم إطلاق الحدث من أحد عناصر التحكم التي تحويها.

ملاحظة:عند استخدام هذه الطريقة سيتم تحديث جميع قيم عناصر التحكم ضمن جميع عناصر UpdatePanel لكن لن يظهر هذا

التغيير إلا ضمن عنصر UpdatePanel الذي أطلق الحدث (من خلال أحد عناصر التحكم المحتواة ضمنه).
تتضمن هذه المقاربة مشكلة تظهر عندما تستلزم أحد التحديثات وقت طويل مما يتيح المجال لمقاطعته بتحديث آخر.
كما نعلم تستخدم ASP.NET إعادة الإرسال غير المتزامن في AJAX والذي يتيح عملية إرسال حتى في حال كون عملية الإرسال السابقة لم تنتهي بعد.
في حال حصول عملية تحديث متزامنة على نفس العنصر ستقوم ASP.NET بمقاطعتها للمحافظة على استقرار و تجانس معلومـــات الجلسة، الكعكات وبيانات حالة التطبيق بشكل عام.
لذلك عند إعادة الإرسال غير المتزامن تقوم ASP.NET بإهمال الطلب القديم وهنا قد تتولد المشكلة.
عند الرغبة في منع مقاطعة عملية إعادة إرسال غير متزامنة يمكننا إضافة رماز JavaScript لإلغاء تفعيل عناصر التحكم في الفتـــرة التي سيكون فيها الطلب قيد الإرسال.
يمكن أيضاً أن يستخدم عنصر تحكم UpdateProgress والذي سيتم تناوله لاحقاً ضمن هذه الجلسة لهذا الغرض.

القادحات

استخدمت الأمثلة التي قمنا باستعراضها في هذه الجلسة السلوك المبيّت في عنصر تحكم UpdatePanel.
ماذا لو أردنا الحصول على تفاعل أكثر تعقيداً، سنضطر إلى المزيد من التحكم بتحديث عناصر UpdatePanel عبر الرماز باستخدام القادحات.
سنقوم في هذا الجزء من الجلسة بإعادة العمل على مثال بطاقة التهنئة الالكترونية الذي يقوم بالتحكم بشكل إظهار وتفاصيل البطاقة.
إن أهم ما يميز هذه المقاربة عن سابقتها هي الحصول على نفس النتيجة باستجابة أعلى.
يتم في هذا المثال استخدام عنصر تحكم UpdatePanel للحصول على الاستجابة المطلوبة.
الطريقة الأسهل للوصول إلى هذا هي إضافة عنصر تحكم ScriptManager ثم وضع كامل الصفحة ضمن عنصر تحكم UpdatePanel واحد كما يوضح الرماز التالي:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<!-- These are the controls for creating the greeting card. -->
<div style="...">
Choose a background color:<br />
<asp:DropDownList id="lstBackColor" runat="server"
Width="194px" AutoPostBack="True">
</asp:DropDownList>
<br /><br />
Choose a foreground (text) color:<br />
<asp:DropDownList id="lstForeColor" runat="server"
Height="22px" Width="194px" AutoPostBack="True" >
</asp:DropDownList>
<br /><br />
Choose a font name:<br />
<asp:DropDownList id="lstFontName" runat="server"
Height="22px" Width="194px" AutoPostBack="True">
</asp:DropDownList>
<br /><br />
Specify a font size:<br />
<asp:TextBox id="txtFontSize" runat="server"
AutoPostBack="True">
</asp:TextBox>
<br /><br />
Choose a border style:<br />
```

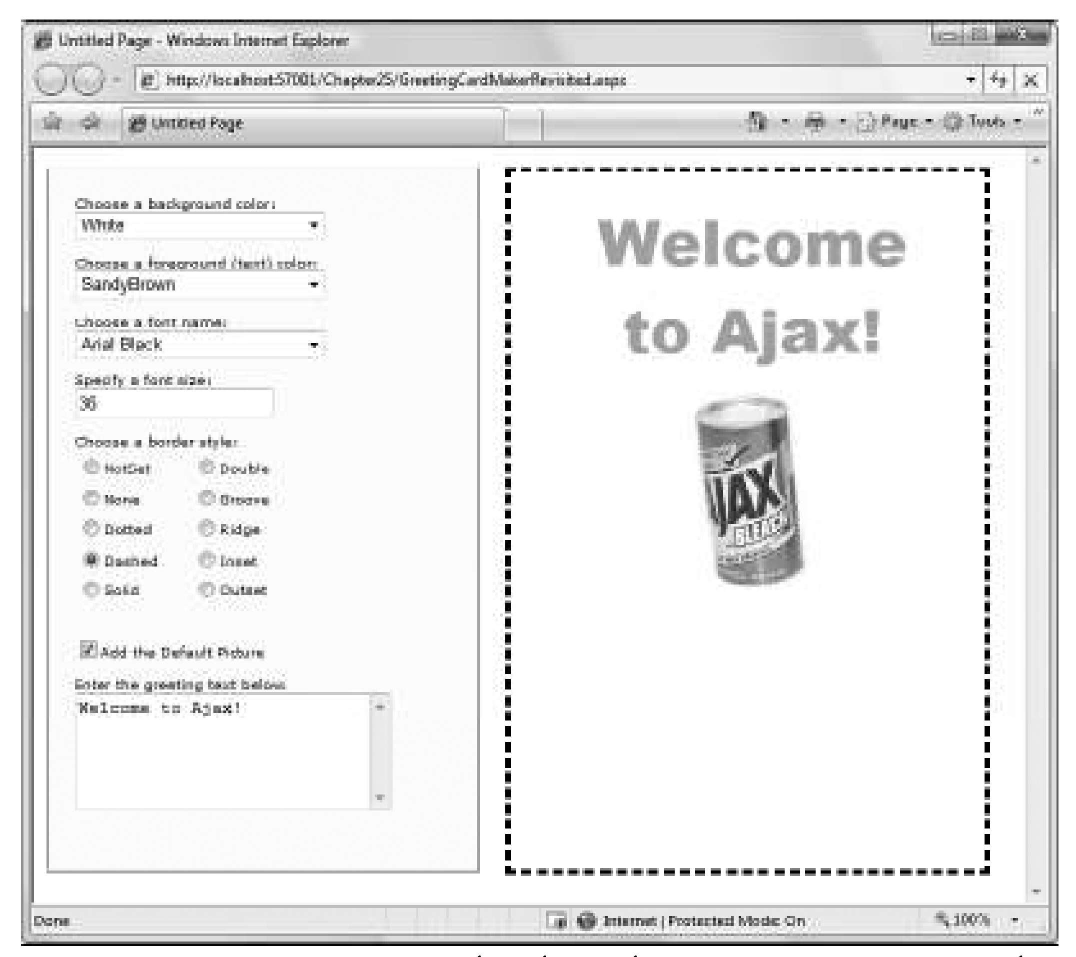
```
<asp:RadioButtonList id="lstBorder" runat="server"
Height="59px" Width="177px" Font-Size="X-Small"
AutoPostBack="True" RepeatColumns="2">
</asp:RadioButtonList>
<br /><br />
<asp:CheckBox id="chkPicture" runat="server"
Text="Add the Default Picture" AutoPostBack="True">
</asp:CheckBox>
<br /><br />
Enter the greeting text below:<br />
<asp:TextBox id="txtGreeting" runat="server"
Height="85px" Width="240px" TextMode="MultiLine"
AutoPostBack="True">
</asp:TextBox>
</div>
<!-- This is the panel that shows the greeting card. -->
<asp:Panel ID="pnlCard" runat="server" ... >
<asp:Label id="lblGreeting" runat="server" Width="272px"
Height="150px"></asp:Label>
<br />
<asp:Image id="imgDefault" runat="server" Width="212px" Height="160px"
Visible="False"></asp:Image>
</asp:Panel>
</ContentTemplate>
</asp:UpdatePanel>
The greeting card is then generated when the Page.Load event fires:
protected void Page_Load(object sender, EventArgs e)
{
if (!this.IsPostBack)
{
// (Initialize all the controls here.)
}
else
{
// Refresh the greeting card.
UpdateCard();
}
}
```

يقوم عنصر التحكم UpdatePanel بمراقبة عناصر التحكم المحتواة ومقاطعة أي طلب يمكن أن يتسبب بإعادة إرسال الصفحة.

يمكن أن يطلق طلب إعادة الإرسال من نقر زر أو كما في مثالنا من أحداث TextBox.TextChanged و ListBox.SelectedItemChanged.

عند تعديل الإعدادات في الجزء الأيسر سيتم تعديل الشكل في القسم الأيمن دون حدوث إعادة إرسال.

في حال إجراء أكثر من تعديل بتتالي سريع سيتم استخدام النتيجة الأخيرة المعادة و الشاملة لجميع التغييرات الحاصلة.



مع أن هذا المثال يؤدي الهدف المطلوب إلا أنه يقوم بأعمال أكثر من اللازم لاتمام هذا الهدف.

تم في هذه المقاربة وضع كامل الصفحة في عنصر تحكم UpdatePanel واحد أي سيتم تحديث نص HTML لكامل الصفحة في كل عملية تحديث.

في هذه الحالة يتمثل الخيار الأفضل بتضمين عناصر البطاقة فقط ضمن عنصر UpdatePanel.

لن تعمل هذه المقاربة بالشكل الصحيح لأنه لن تتم عملية اعتراض أي من الأحداث التي تنتجها عناصر التحكم الغير محتواة في UpdatePanel و ستتسبب في إعادة إرسال كامل الصفحة.

يكمن الحل في تكليف عنصر UpdatePanel بشكل صريح بمراقبة الأحداث التي تصدرها عناصر تحكم معينة حتى ولو لم تكن محتواة ضمن عنصر UpdatePanel هذا.

تتم هذه العملية بإضافة قادحات إلى عنصر التحكم UpdatePanel.

يمكن إضافة قادح واحد لكل زر كما يبين الرماز التالي:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
<!-- The controls for creating the greeting card go here. -->
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<!-- This is the panel that shows the greeting card. -->
<asp:Panel ID="pnlCard" runat="server" ... >
<asp:Label id="lblGreeting" runat="server" Width="272px"
Height="150px"></asp:Label>
<asp:Image id="imgDefault" runat="server" Width="212px" Height="160px"
Visible="False"></asp:Image>
</asp:Panel>
</ContentTemplate>
<Triggers>
<asp:AsyncPostBackTrigger ControlID="lstBackColor" />
```

```
<asp:AsyncPostBackTrigger ControlID="lstForeColor" />
<asp:AsyncPostBackTrigger ControlID="lstFontName" />
<asp:AsyncPostBackTrigger ControlID="txtFontSize" />
<asp:AsyncPostBackTrigger ControlID="lstBorder" />
<asp:AsyncPostBackTrigger ControlID="chkPicture" />
<asp:AsyncPostBackTrigger ControlID="txtGreeting" />
</Triggers>
</asp:UpdatePanel>
```

تقوم القادحات بإبلاغ عنصر UpdatePanel بمقاطعة الحدث التلقائي لعناصر التحكم تلك.

يمكن استخدام القادحات بطريقة أخرى غير مراقبة بعض عناصر التحكم يمكن أيضاً استخدامها لإهمال الطلبات غير المتزامنة واستخدام طلب إعادة إرسال كامل للصفحة عوضاً عنه.

نستخدم لهذا الغرض القادح PostBackTrigger كما هو موضح في الرماز التالي:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
<ContentTemplate>
<asp:Label ID="Label1" runat="server" Font-Bold="True"></asp:Label>
<br />
<br />
<asp:Button ID="cmdPostback" runat="server" Text="Refresh Full Page" />
</ContentTemplate>
<Triggers>
<asp:PostBackTrigger ControlID="cmdPostback" />
</Triggers>
</asp:UpdatePanel>
```

لا تعتبر هذه التقنية شائعة الاستخدام و لكنها قد تكون مفيدة جداً في بعض الأحيان.

ذكرنا سابقاً ضمن هذه الجلسة أن عنصر تحكم UpdatePanel يعمل في الخلفية بشكل غير متزامن مما يسمح للمستخدم باستمرار العمل على الصفحة.

المشكلة تكمن في الزمن الذي قد تأخذه أحد الطلبات غير المتزامنة. في هذه الحالة قد يظن المستخدم أن التطبيق لا يتجاوب ويقوم بإعادة ضغط الزر لأكثر من مرة.

تتسبب هذه المشكلة بتوليد عمل إضافي وعبء على التطبيق وبالتالي تزيد من بطئ التطبيق أكثر فأكثر.

يتمثل الحل الذي توفره Asp.net في عنصر تحكم UpdateProgress.

يعمل عنصر التحكم هذا مع عنصر تحكم UpdatePanel. ويمكّن من إظهار رسالة في أثناء تخديم الطلبات غير المتزامنة التي تستهلك وقتاً طويلاً نسبياً.

إظهار محاكاة لشريط التقدم

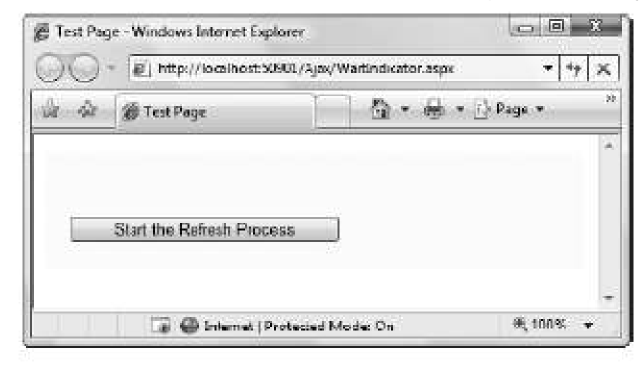
يمكننا عند إضافة عنصر تحكم UpdateProgress تحديد مكونات تظهر حال بدء الطلب الغير متزامن و تختفي عند إنتهائه.

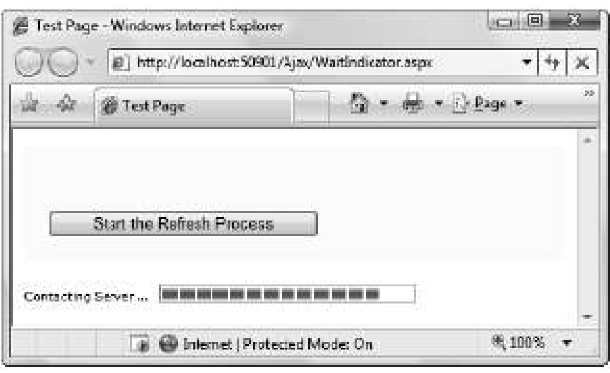
تتضمن المكونات الممكن إظهارها، الصور الثابتة أو المتحركة من نمط animated GIF لأنها توحي بشكل فعال أكثر أن الصفحة ما تزال تعمل.

يظهر الشكل التالي صفحة تقوم باستخدام عنصر تحكم UpdateProgress في ثلاث نقاط من دورة عملها.

تمثل المشاهد التالية وحسب شكل الصفحة الحال التي ستظهر عليها الصفحة قبل تلقي الطلب غير المتزامن و عند بداية تنفيذ الطلب

وعند الانتهاء من التنفيذ.







النص البرمجي لعمل هذا التطبيق هو على الشكل:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<div style="background-color:#FFFFE0;padding: 20px">
<asp:Label ID="lblTime" runat="server" Font-Bold="True"></asp:Label>
<br /><br />
<asp:Button ID="cmdRefreshTime" runat="server"
OnClick="cmdRefreshTime_Click"
Text="Start the Refresh Process" />
</div>
</ContentTemplate>
</asp:UpdatePanel>
<br />
<asp:UpdateProgress runat="server" id="updateProgress1">
<ProgressTemplate>
<div style="font-size: xx-small">
Contacting Server ... <img src="wait.gif" />
</div>
</ProgressTemplate>
</asp:UpdateProgress>
```

لا تعتبر هذه هي الطريقة الوحيدة لاستخدام عنصر التحكم UpdateProgress إذ يمكن واعتماداً على التوزيع المطلوب إضافة عنصر

تحكم UpdateProgress ضمن عنصر تحكم UpdatePanel.

يتميز النص البرمجي في هذه الحالة بفارق طفيف عن الحالات السابقة يتمثل بتمكين إظهار محتوى العنصر UpdateProgress فقط عند كون الطلب قيد التنفيذ.

يمكننا إضافة أسطر إضافية لمحاكاة عملية بطء التنفيذ و ذلك كما يلي:

```
protected void cmdRefreshTime_Click(object sender, EventArgs e)
{
System.Threading.Thread.Sleep(TimeSpan.FromSeconds(10));
lblTime.Text = DateTime.Now.ToLongTimeString();
}
```

لا حاجة لإجراء عملية ربط خاصة لعنصر التحكم UpdatProgress مع عنصر تحكم UpdatePanel حيث يقوم عنصر تحكم UpdateProgress بإظهار محتوى القالب ProgressTemplate عند مباشرة أي عنصر UpdatePanel بطلب غير متزامن.

على كل حال عند العمل ضمن صفحة معقدة تحتوي أكثر من عنصر تحكم UpdatePanel يمكننا التحكم بحصر استجابة عنصر تحكم UpdateProgress بواحد من هذه العناصر فقط.

يكفي لتطبيق هذه المقاربة إسناد قيمة معرف ID الخاص بعنصر UpdatePanel إلى الخاصة UpdateProgress.AssociatedUpdatePanelID. حتى أنه من الممكن ضمن هذا السياق إضافة أكثر من عنصر تحكم UpdateProgress وربط كل منها بعنصر تحكم UpdatePanel مختلف.

الإلغاء

توفر عناصر تحكم UpdateProgress إمكانية تحديد زر إلغاء.

عند نقر المستخدم على هذا الزر ستتم عملية إلغاء فوري للطلب غير المتزامن، وسيتم إخفاء محتوى العنصر UpdateProgress وتعود الصفحة إلى حالتها الأصلية.

إن عملية إضافة زر إلغاء هي إجرائية بمرحلتين:

1. إضافة نص JavaScript بشكل حرفي إلى نهاية رماز الصفحة بعد جميع المكونات وقبل إغلاق التأشيرة </body> كما يظهر في الرماز التالي:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="WaitIndicator.aspx.cs"
Inherits="WaitIndicator" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
...
</head>
<body>
<form id="form1" runat="server">
...
</form>
<script type="text/javascript">
var prm = Sys.WebForms.PageRequestManager.getInstance();
prm.add_initializeRequest(InitializeRequest);
function InitializeRequest(sender, args)
{
if (prm.get_isInAsyncPostBack())
{
args.set_cancel(true);
}
}
function AbortPostBack()
{
if (prm.get_isInAsyncPostBack()) {
```

```
prm.abortPostBack();
}
}
</script>
</body>
</html>
```

2. بعد إضافة هذا النص البرمجي يمكننا استدعاء الطريقة ()AbortPostBack في أي وقت لإلغاء عملية الإرسال غير المتزامن.

تعتبر أسهل طريقة للوصول إلى هذا الهدف وصل حدث JavaScript بالطريقة ()AbortPostBack باستخدام واصفة حدث JavaScript.

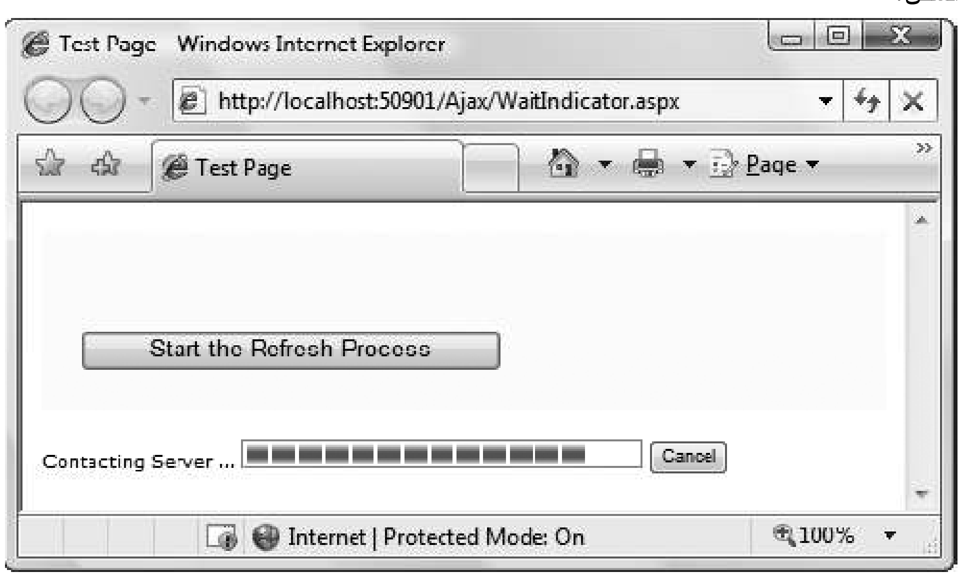
يمكننا إضافة واصفة حدث JavaScript لأي عنصر HTML. فعلى سبيل المثال يمكن التعامل مع حدث النقر من جهة الزبون باستخدام الواصفة onclick.

يستخدم الرماز التالي هذه التقنية لربط زر مع التابع ()AbortPostBack.

```
<input id="cmdCancel" onclick="AbortPostBack()" type="button" value="Cancel" />
```

عند النقر على هذا الزر سيتم إطلاق التابع ()AbortPostBack وسيتم إيقاف عمليات الاستدعاء غير المتزامن فوراً.

الاستخدام النمطي لمثل زر الإلغاء هذا هو ضمن ProgressTemplate الخاص بعنصر تحكم UpdateProgress كما يظهر في الشكل:



التحديث وفق فاصل زمني

يحتاج المستخدم أحياناً إلى تمكين عملية تحديث الصفحة دون انتظار أي فعل من المستخدم.

قد نحتاج على سبيل المثال إلى إنشاء صفحة خاصة بمؤشرات الأسهم حيث لا بد من تحديث قيم المؤشرات بشكل دوري (كل خمسة دقائق مثلاً). يمكن في هذه الحالة استخدام عنصر التحكم Timer الذي توفره ASP.NET و الذي يسمح بتنفيذ هذا التصميم بسهولة.

يكفي لاستخدام عنصر التحكم هذا إضافته إلى الصفحة وتحديد فاصل التحديث كما يلي:

```
<asp:Timer ID="Timer1" runat="server" Interval="60000" />
```

ملاحظة: لا بد من التفكير ملياً قبل استخدام عنصر التحكم Timer ضمن التطبيق لما يضيفه من عبء على التطبيق وتأثيره على تقليل قابلية التوسيع كذلك يجب محاولة إبقاء زمن التحديث أطول ما يمكن.

يقوم المخدم بإطلاق حدث Tick والذي يمكن معالجته لتحديث الصفحة.

يمكن أيضاً الاعتماد على أحداث أخرى مثل Page.Load، لأن عنصر التحكم Timer يقود إلى إطلاق كامل دورة حياة الصفحة مع كل ظهور لحدث Tick.

يناسب عنصر التحكم Timer الصفحات التي تستخدم التصيير الجزئي.

لاستخدام عنصر التحكم Timer مع التصيير الجزئي نقوم بتضمين الجزء المطلوب تحديثه من الصفحة في عنصر تحكم UpdatePanel مع الانتباه إلى إسناد القيمة Conditional إلى الخاصة UpdateMode ثم إضافة قادح لقسر عملية التحديث مع كل ظهور للحدث Timer.Tick كما يوضح الرماز التالي:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
<ContentTemplate>
...
</ContentTemplate>
<Triggers>
<asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
</Triggers>
</asp:UpdatePanel>
```

ملاحظة: لا بد من استعمال القادحات مع عنصر تحكم Timer. لن تتمكن ببساطة من وضع هذا العنصر ضمن عنصر تحكم UpdatePanel لأنه سيقوم في هذه الحالة بإعادة إرسال الصفحة بشكل كامل مع ارتجاف في الصفحة.

لإيقاف عنصر تحكم Timer يمكنك ببساطة إسناد القيمة false إلى الخاصة Enabled في الرماز من جهة المخدم.

يبين الرماز التالي كيف تتم عملية إلغاء تفعيل عنصر التحكم Timer بعد عشر تحديثات:

```
protected void Timer1_Tick(object sender, EventArgs e)
{
// Update the tick count and store it in view state.
int tickCount = 0;
if (ViewState["TickCount"] != null)
{
tickCount = (int)ViewState["TickCount"];
}
tickCount++;
ViewState["TickCount"] = tickCount;
// Decide whether to disable the timer.
if (tickCount > 10)
{
Timer1.Enabled = false;
}
}
```

عناصر تحكم أدوات Ajax في ASP.NET

مجموعة أدوات AJAX عدا عناصر تحكم UpdatePanel، UpdateProgress وTimer هي مشروع مشترك بين Microsoft وASP.NET Community.

يتضمن هذا المشروع عشرات عناصر التحكم التي تستخدم مكتبات ASP.NET AJAX لإنشاء تأثيرات معقدة.

اتجه العديد من المطورين إلى استخدام هذه المكتبات لعدة أسباب:

المجانية.

تتضمن كامل النص البرمجي لتمكين المطورين الطموحين من إجراء التعديلات على أي عنصر تحكم.

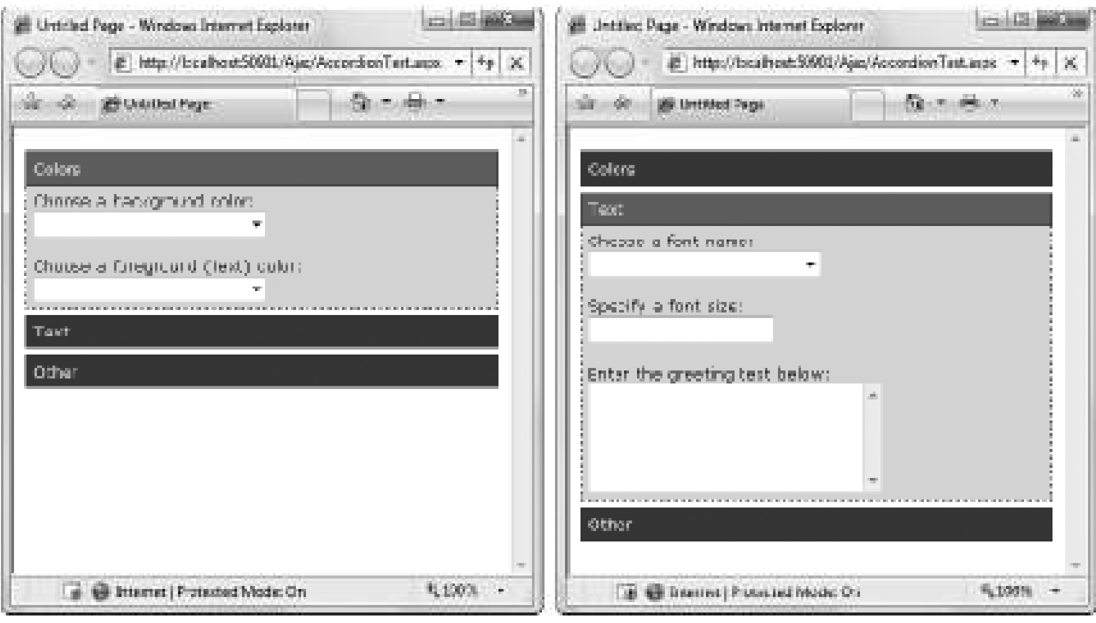
تستخدم موسعات لتحسين عمل عناصر تحكم الوب في ASP.NET.

مثال استخدام عنصر تحكم Accordion:

عنصر تحكم Accordion هو عبارة عن حاوية تقوم بمراكمة مجموعة من اللوحات فوق بعضها البعض وتمكّن رؤية محتوى أحدها فقط في وقت واحد.

لكل لوحة من هذه اللوحات ترويسة. عند ضغط هذه الترويسة يتم عرض محتوى اللوحة.

يوضح الشكل التالي كيف يبدو عنصر تحكم Accordion مؤلفاً من ثلاث لوحات (Colors, Text, Other).



في المرة الأولى التي يتم فيها تحميل الصفحة يتم تحميل جميع اللوائح وتصييرها إلى لغة HTML ولكن يتم إخفاؤها باستخدام أنماط CSS.

عند نقر الترويسة يقوم تابع JavaScript بالعمل وتغيير واصفات هذه الأنماط لإظهار محتوى اللوحة التي تم النقر على ترويستها وإخفاء محتوى اللوحة السابقة.

يمكن باسناد القيمة true إلى الخاصة FadeTransition إعطاء مظهر أكثر إنسيابية لحركة الإظهار تلك.

النص التالي يوضح البنية التي يتم استعمالها لوضع عنصري AccordionPane ضمن عنصر تحكم Accordion:

```
<ajaxToolkit:Accordion ID="Accordion1" runat="server">
<Panes>
<ajaxToolkit:AccordionPane runat="server">
...
</ajaxToolkit:AccordionPane>
<ajaxToolkit:AccordionPane runat="server">
...
</ajaxToolkit:AccordionPane>
</Panes>
</ajaxToolkit:Accordion>
```

ملاحظة:

- يمكننا استخدام الخاصة Accordion.SelectedIndex لتحديد اللوحة التي سيتم إظهارها من لوائح عنصر تحكم

Accordion

- في حال إسناد القيمة true إلى الخاصة RequiredOpenedPane سيتم إظهار محتوى أحد اللوائح بشكل تلقائي أي لن نتمكن من إغلاق جميع اللوائح في آن معاً

يتكون كل عنصر تحكم AccordionPane من مقطعين أساسيين:

- الرأس: يستخدم لإظهار اسم اللوحة وكزر قابل للضغط لتأمين فتح و إغلاق اللوحة.
- المحتوى: يحمل هذا الجزء العناصر التي تود تضمينها في هذه اللوحة.

فيما يلي الرماز الواجب استخدامه للحصول على الواجهة المبينة في الشكل رقم 5.

```
<ajaxToolkit:Accordion ID="Accordion1" runat="server"
HeaderCssClass="accordionHeader"
HeaderSelectedCssClass="accordionHeaderSelected"
ContentCssClass="accordionContent">
<Panes>
<ajaxToolkit:AccordionPane runat="server">
<Header>Colors</Header>
<Content>
Choose a background color:<br />
<asp:DropDownList id="lstBackColor" runat="server"
Width="194px" AutoPostBack="True">
</asp:DropDownList>
<br /><br />
Choose a foreground (text) color:<br />
<asp:DropDownList id="lstForeColor" runat="server"
Height="22px" Width="194px" AutoPostBack="True" >
</asp:DropDownList>
</Content>
</ajaxToolkit:AccordionPane>
<ajaxToolkit:AccordionPane runat="server">
<Header>Text</Header>
<Content>
Choose a font name:<br />
<asp:DropDownList id="lstFontName" runat="server"
Height="22px" Width="194px" AutoPostBack="True">
</asp:DropDownList>
<br /><br />
Specify a font size:<br />
<asp:TextBox id="txtFontSize" runat="server"
AutoPostBack="True">
</asp:TextBox>
<br /><br />
Enter the greeting text below:<br />
<asp:TextBox id="txtGreeting" runat="server"
Height="85px" Width="240px" TextMode="MultiLine"
AutoPostBack="True">
</asp:TextBox>
</Content>
</ajaxToolkit:AccordionPane>
<ajaxToolkit:AccordionPane runat="server">
<Header>Other</Header>
<Content>
Choose a border style:<br />
<asp:RadioButtonList id="lstBorder" runat="server"
Height="59px" Width="177px" Font-Size="X-Small"
AutoPostBack="True" RepeatColumns="2">
</asp:RadioButtonList>
<br /><br />
<asp:CheckBox id="chkPicture" runat="server"
Text="Add the Default Picture" AutoPostBack="True">
</asp:CheckBox>
</Content>
</ajaxToolkit:AccordionPane>
</Panes>
```

```
</ajaxToolkit:Accordion>
```

عنصر التحكم AutoCompleteExtender

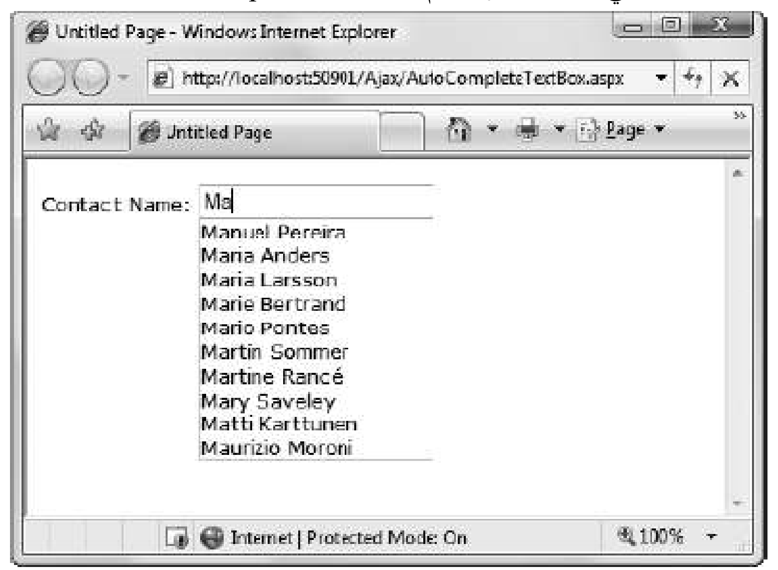
ليست جميع عناصر تحكم أدوات AJAX في ASP.NET هي عناصر تحكم جديدة كلياً مثل عنصر التحكم Accordion بل هي في معظمها موسعات لإضافة مزايا جديدة لعناصر تحكم ASP.NET التقليدية.

تمكّن هذه الموسعات من استخدام نفس التأثيرات على عدد كبير من عناصر التحكم.

تصبح هذه المقاربة مفيدة جداً عندما نتعامل مع ميزة متعددة الأغراض مثل الإتمام الآلي **AutoCompleteExtender**.

يمكّن عنصر التحكم AutoCompleteExtendere من إظهار قائمة بالاقتراحات أثناء تحرير المستخدم لمحتوى أحد عناصر التحكم النصية.

يبين الشكل التالي شكل عنصر تحكم AutoCompleteExtender أثناء العمل مع عنصر تحكم TextBox عادي.



عندما يبدأ المستخدم بالكتابة ضمن العلبة النصية تظهر قائمة تقدم مجموعة من الاقتراحات.

عندما ينقر المستخدم على أحد هذه الخيارات من القائمة يتم نسخ نص هذا الخيار إلى العلبة النصية.

يلزمنا لتطبيق هذا المثال عنصر تحكم علبة نصية عادية:

```
Contact Name:<asp:TextBox ID="txtName" runat="server"></asp:TextBox>
```

بعد إضافة العلبة النصية يجب إضافة عنصري تحكم ScriptManager و AutoCompleteExtender. يتم الاستحصال على قائمة الاقتراحات التي يقدمها عنصر التحكم AutoCompleteExtender من إجرائية خاصة سنحتاج لإنشائها ضمن صفحتنا.

لربط عنصر التحكم AutoCompleteExtender بعنصر تحكم ما نستخدم الخاصة TargetControlID حيث نقوم بإسناد قيمة ID الخاصة بعنصر التحكم المراد الربط معه إلى هذه الخاصة.

تحدد الخاصة MinimumPrefixLength عدد المحارف التي يجب كتابتها قبل تقديم أي إقتراح لإظهار قائمة الإتمام الآلي.

تحدد الخاصة ServiceMethod اسم طريقة الوب التي سيتم استخدامها للحصول على قائمة الاقتراحات.

```
<ajaxToolkit:AutoCompleteExtender ID="autoComplete1" runat="server"
TargetControlID="txtName" ServiceMethod="GetNames" MinimumPrefixLength="2">
</ajaxToolkit:AutoCompleteExtender>
```

لا بد من إنشاء طريقة الوب GetNames ليعمل هذا المثال بنجاح.

```
[System.Web.Services.WebMethod]
[System.Web.Script.Services.ScriptMethod]
public static List<string> GetNames(string prefixText, int count)
{ ... }
```

تقبل طريقة الوب المعرّفة معاملين يحددان النص الذي كتبه المستخدم وعدد المقترحات المطابقة المراد إظهارها (القيمة الافتراضية هي 10 لهذا المعامل).

تحدد الواصفتان السابقتان للطريقة GetNames أن هذه الطريقة هي طريقة وب بمعنى أن الزبون يستطيع استدعاءها بشكل مباشر باستعمال طلبات HTTP وبأنها تدعم الاستدعاء باستخدام JavaScript (وهي الطريقة التي يستخدمها عنصر AutoCompleteExtender).

كتابة النص البرمجي الذي سيقوم بالاستحصال على المقترحات قد يكون مرهقاً بعض الشيء، سنستخدم في مثالنا قائمة من الأسماء يتم الاستحصال عليها من قاعدة البيانات NorthWind.

للتأكد من أن هذه العملية ستتم لمرة واحدة وليس كل مرة يتم فيها التعامل مع عنصر مرتبط مع AutoComleteExtender، نلجأ عادة إلى استخدام الذاكرة الخبيئة كما يوضح الرماز التالي:

```
List<string> names = null;
// Check if the list is in the cache.
if (HttpContext.Current.Cache["NameList"] == null)
{
// If not, regenerate the list. The ADO.NET code for this part
// isn't shown (but you can see it in the downloadable examples
// for this chapter.
names = GetNameListFromDB();
// Store the name list in the cache for sixty minutes.
HttpContext.Current.Cache.Insert("NameList", names, null,
DateTime.Now.AddMinutes(60), TimeSpan.Zero);
}
else
{
// Get the name list out of the cache.
names = (List<string>)HttpContext.Current.Cache["NameList"];
}
...
```

المرحلة التالية في هذا النص هي مطابقة عناصر القائمة لإيجاد العناصر التي تبدأ بنفس الأحرف التي تمت كتابتها:

```
...
int index = -1;
for (int i = 0; i < names.Count; i++)
{
// Check if this is a suitable match.
if (names[i].StartsWith(prefixText))
{
index = i;
break;
}
}
// Give up if there isn't a match.
if (index == -1) return new List<string>();
...
```

بعده يبدأ الرماز الخاص بالبحث بدليل الموقع الحالي وينتقل عبر اللائحة ويحاول الاستحصال على عشرة اقتراحات. فإما أن يصل إلى نهاية القائمة أو يجد قيمة لا تطابق السابقة التي تم إدخالها فيوقف البحث.

```
...
List<string> wordList = new List<string>();
for (int i = index; i < (index + count); i++)
{
// Stop if the end of the list is reached.
if (i >= names.Count) break;
// Stop if the names stop matching.
```

```
if (!names[i].StartsWith(prefixText)) break;
wordList.Add(names[i]);
}
...
```

وأخيراً تتم إعادة جميع النتائج المطابقة.

```
...
return wordList;
```

الحصول على المزيد من عناصر تحكم أدوات AjAX

عناصر Accordion و AutoCompleteExtender هي أمثلة بسيطة عما تحتويه مكتبة أدوات ASP. NET العناصر والتي تحتوي على أكثر من 30 عنصر مكون.

الطريقة الأسهل للبدء باستكشاف عناصر التحكم الأخرى قم بزيارة الموقع http://ajax.asp.net/ajaxtoolkit حيث يمكن إيجاد مراجع تصف كل عنصر و طريقة عمله و تمكنك من اختباره مباشرة عن طريق الموقع.

من أهم عناصر التحكم المفيد استكشافها:

AlwaysVisibleControlExtender, AnimationExtender, CalendarExtender, DragPanelExtender, DynamicPopulateExtender, FilteredTextBoxExtender, HoverMenuExtender, ListSearchExtender, ModalPopupExtender, MutuallyExclusiveCheckboxExtender NumericUpDownExtender, PagingBulletedListExtender, PasswordStrengthExtender, PopupControlExtender, RatingResizableControlExtender, SlideShowExtender, TabContainer, TextBoxWatermark

الفصل الثالث عشر

الخبء في ASP.NET

يتعرف الطالب في هذه الجلسة على:

1- مفهوم الخبء في تطبيقات الوب
2- متى نستخدم الخبء
3- أنواع الخبء التي تدعمها ASP.NET و كيفية استخدامها

الخبء

تختلف تطبيقات الوب بشكل عام عن تطبيقات سطح المكتب كونها تحتاج إلى متطلبات خاصة مثل الحاجة لتخديم عدد كبير من الزبائن بشكل سريع و بسيط.

تغري بيئة ASP.NET المطورين باستخدام العديد من الميزات التي قد تقود أحياناً إلى نسيان المطور أنه يعمل على تطبيق وب.

يؤدي استخدام هذه التقنيات أو الآليات إلى إبطاء أو حتى إعاقة عمل التطبيق.

لحسن الحظ يمكن أن نبني تطبيق يستفيد من المميزات الكثيرة التي تمنحها ASP.NET مثل طرق إدارة الحالة المختلفة وعناصر تحكم الوب.

سيحتاج المطور للمزيد من الوقت في عملية ضبط الأداء للموقع.

أحد أسهل الطرق لتحسين الأداء هي الخبء. في عملية الخبء يتم تخزين المعلومات القيمة بشكل مؤقت على ذاكرة المخدم ليتم إعادة استخدامها لاحقاً.

إن عملية الخبء و بعكس ما رأيناه في الطرق المختلفة لإدارة الحالة تتضمن آلية مبيتة تكفل الأداء الجيد للتطبيق.

الخبء في ASP.NET

تمت العديد من التطويرات على الخبء في ASP.NET منذ النسخ الأولى حيث تمت زيادة فعالية المزايا المضمنة في هذه الآلية بشكل **يرفع الأداء في** بعض السيناريوهات بشكل كبير.

يستخدم الخبء عادة لتخزين المعلومات التي يتم الاستحصال عليها من قاعدة البيانات لغرض توفير الوقت اللازم لإعادة استعادتها في كل مرة.

يمكن بعملية ضبط للأداء باستخدام الخبء تخفيض العبء للحد الأدنى، حيث سيتم الحصول على بعض المعلومات المطلوبة من ذاكرة الخبء عوضاً عن إعادة الاستعلام عنها من قاعدة البيانات.

ليست عملية تخزين المعلومات ضمن الذاكرة دائماً فكرة جيدة. إن ذاكرة المخدم هي مورد محدود، ستؤدي المبالغة في استخدامها إلى تخزين الأجزاء الفائضة على القرص الصلب مما سيقود النظام كاملاً إلى حالة من انخفاض في الأداء.

لهذا تم العمل على جعل تقنية الخبء فيASP.NET ذاتية المحدودية. حيث غالباً ما يتم توفير المعلومات التي يتم خبؤها ضمن ذاكرة

المخدم.

يظل زمن الاحتفاظ بهذه المعلومات ضمن إدارة المخدم. فإذا امتلأت ذاكرة المخدم أو حصل استهلاك كبير للذاكرة من قبل التطبيقات العاملة ستتم عملية إخلاء اصطفائية لذاكرة الخبء وستتابع التطبيقات عملها بأداء جيد.

متى نستخدم الخبء

يكمن السر الأساسي في الاستفادة الأعظمية من الخبء في اختيار الزمن المناسب لاستخدامها.

من استراتيجيات الخبء الناجح التركيز على تخزين المعلومات الأكثر استخداماً و التي تستهلك زمن كبير نسبياً لإنشائها وتخزينها.

إن خبء معلومات غير مهمة نسبياً قد يؤدي إلى إجبار إخلاء معلومات أكثر أهمية.

هناك نقطتين أساسيتين ينصح بمراعاتهما عند استخدام الخبء:

قم بخبء البيانات عالية الكلفة.

قم بخبء البيانات المستخدمة بكثرة.

يمكن بمراعاة هاتين النقطتين الوصول إلى أداء أعلى و قابلية أكبر للتوسع.

الخبء في ASP.NET

للخبء في ASP.NET نوعان أساسيان:

- خبء الخرج: وهو أبسط أشكال الخبء. يتم الاحتفاظ في هذا النوع من الخبء بنسخة عن صفحة HTML التي تم تصييرها وإرسالها إلى الزبون.
- خبء البيانات: تتم هذه العملية يدوياً ضمن الرماز الذي يكتبه المطور. حيث يتم تخزين الأجزاء الهامة من المعلومات التي تستهلك وقتاً لإنشائها. يعتبر خبء البيانات من حيث المفهوم مشابهاً لحالة التطبيق لكنه يعتبر حل أفضل بكثير من وجهة نظر التأثير على أداء المخدم.

هناك أيضاً نوعان خاصان من الخبء يمكن استخدامهما بالاعتماد على النوعين الرئيسيين أعلاه:

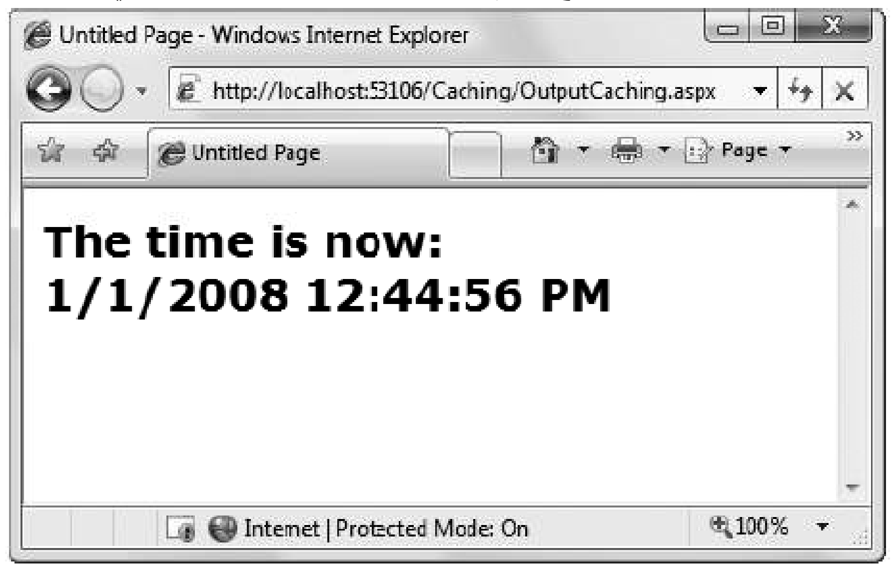
- الخبء الجزئي: يعد هذا النوع من الخبء حالة خاصة من خبء الخرج حيث يتم تخزين أجزاء من خرج صفحة HTML المرسل إلى الزبون عوضاً عن خبء كامل الصفحة.
- خبء مصادر البيانات: يعبر هذا النوع من الخبء عن القدرة المبيتة لدى عناصر تحكم البيانات مثل SqlDataSource وXMLDataSource وObjectDataSource على خبء البيانات. يستخدم هذا النوع من الخبء ،عملياً، طريقة خبء البيانات. يكمن الاختلاف في أن المطور ليس بحاجة أن يعالج العملية بشكل صريح إذ يكفي إعداد الخصائص المناسبة ليقوم عنصر تحكم مصدر البيانات بإدارة عملية خبء، تخزين واستعادة البيانات تلقائياً.

خبء الخرج

يتم في هذا النوع كما ذكرنا خبء كامل صفحة HTML و عند طلب الصفحة مرة أخرى لن تتم إعادة إنشاء أغراض عناصر التحكم

أو إعادة دورة حياة الصفحة من جديد كما لن يتم تنفيذ أي رماز بل سيتم إعادة إرسال النسخة المخزنة من الصفحة إلى الزبون.

لنتمكن من رؤية عمل خبء الخرج سنقوم ببناء صفحة بسيطة تظهر الوقت الحالي كما هو موضح في الشكل أدناه:



أما الرماز فهو كما يلي:

```
public partial class OutputCaching : System.Web.UI.Page
{
protected void Page_Load(Object sender, EventArgs e)
{
lblDate.Text = "The time is now:<br />";
lblDate.Text += DateTime.Now.ToString();
}
}
```

الطريقة الأكثر شيوعاً لتفعيل خبء الصفحة: هي إضافة الموجه OutputCache في بداية ملف aspx تحت موجه الصفحة تماماً:

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
```

يستخدم معامل Duration لتحديد الفترة التي سيتم فيها خبء الصفحة و هي 20 ثانية في مثالنا.

المعامل VaryByParam هو معامل إلزامي سنؤجل شرحه إلى مرحلة لاحقة ضمن هذه الجلسة.

عند تشغيل هذا المثال سنلاحظ بأن أي عملية تحديث ضمن فترة لا تتجاوز 20 ثانية سوف لن تؤدي إلى تحديث الوقت و التاريخ الحالي.

إن الزمن الذي سيتم الاحتفاظ خلاله بنسخة عن الصفحة هو 20 ثانية فقط في حال لم يتم إخلاء الصفحة قبل ذلك من ذاكرة الخبء بسبب امتلائها.

الخبء من طرف الزبون

أحد الخيارات المتوفرة للمطور هي الخبء من طرف الزبون.

في هذه الحالة يقوم المستعرض بتخزين نسخة من الصفحة. تستخدم هذه النسخة عند عودة الزبون إلى الصفحة أو عند إعادة كتابة اسمها ضمن شريط العنوان.

سيتم إهمال النسخة المخزنة على الزبون في حال ضغط الزر Refresh وسيعاد طلب الصفحة من المخدم.
يمكن خبء صفحة من طرف الزبون باستخدام الواصفة Location ضمن الموجه OutputCache والتي تحدد قيمة من الترقيم System.Web.UI.OutputCacheLocation كما هو موضح أدناه:

```
<%@ OutputCache Duration="20" VaryByParam="None" Location="Client" %>
```

لا يقدم الخبء من جهة الزبون فائدة مماثلة لتلك التي يحققها الخبء من جهة المخدم لأنه لايمنع بناء نسخ مختلفة لكل مستخدم ولكنها تظل طريقة مفيدة يمكن أن تساهم في تخفيف العبء عن المخدم و بالأخص عند العمل مع التطبيقات التي تعتمد على تخصيص البيانات.

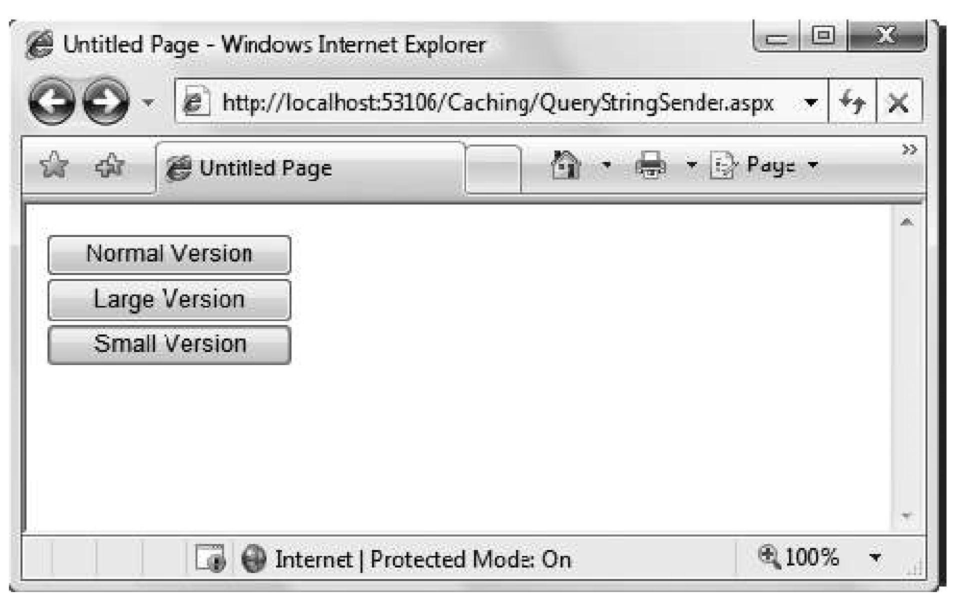
الخبء و سلسلة محارف الاستعلام

أحد أهم الأمور التي يجب مراعاتها في خبء البيانات هي متى يجب إعادة استخدام البيانات المخزنة ومتى يجب إعادة طلبها مرة أخرى.
نستخدم لتسهيل عملية اتخاذ القرار بهذا الخصوص ضمن التطبيق الواصفة VaryByParam ضمن الموجه OutputCache.
تمكن هذه الواصفة ASP.NET من تحديد خبء نسخة واحدة من الصفحة أو مجموعة من النسخ.
يمكن لـــASP.NET تخزين نسخ تختلف عن بعضها البعض بأحد أو جميع المعاملات التي يتم تمريرها ضمن سلسلة محارف الاستعلام (Query String).
استخدمنا في المثال السابق القيمة None للواصفة VaryByParam. تحدد هذه القيمة خبء لنسخة واحدة من الصفحة.
لتمكين عملية خبء نسخة جديدة مع كل تغيير في أي معامل ضمن سلسلة محارف الاستعلام نسند القيمة * إلى الواصفة VaryByParam.
لتحديد معامل أو مجموعة معاملات ليتم أخذها بعين الاعتبار عند خبء نسخة جديدة يكفي وضع أسماء هذه المعاملات مفصولة بفاصلة منقوطة ضمن قيمة الواصفة VarByParam أي من الشكل:

```
<%@ OutputCache Duration="20" VaryByParam="ProductID;CurrencyType" %>
```

مثال:

يستخدم المثال التالي صفحتي وب لتوضيح كيفية خبء أكثر من نسخة من صفحة وب بشكل منفصل.
توفر الصفحة الأولى QueryStringSender.aspx واجهة تحتوي ثلاثة أزرار كما يبين الشكل:



تستخدم الأزرار الثلاثة مقبض معالجة حدث وحيد يقوم بالانتقال إلى الصفحة QueryStringRecipient.aspx مع إضافة معامل يحدد رقم النسخة إلى سلسلة محارف الاستعلام كما في الرماز التالي:

```
protected void cmdVersion_Click(Object sender, EventArgs e)
{
Response.Redirect("QueryStringRecipient.aspx" + "?Version=" +
((Control)sender).ID);
}
```

تستخدم الصفحة QueryStringRecipient.aspx رسالة إظهار التاريخ والوقت التي قمنا باستخدامها في مثالنا السابق ضمن هذه الجلسة مع فارق إسناد القيمة Version إلى الواصفة VaryByParm ضمن موجه OutputCache:

```
<%@ OutputCache Duration="60" VaryByParam="Version" %>
```

لتمييز النسخ المختلفة التي سيتم خبؤها بحسب تغير الواصفة VarByParam سنقوم بتغيير حجم الخط ضمن مقبض حدث تحميل الصفحة بحسب قيمة هذا المعامل Version كما يلي:

```
protected void Page_Load(Object sender, EventArgs e)
{
lblDate.Text = "The time is now:<br />" + DateTime.Now.ToString();
switch (Request.QueryString["Version"])
{
case "cmdLarge":
lblDate.Font.Size = FontUnit.XLarge;
break;
case "cmdNormal":
lblDate.Font.Size = FontUnit.Large;
break;
case "cmdSmall":
lblDate.Font.Size = FontUnit.Small;
break;
}
}
```

التحكم المخصص بالخبء

ليس استخدام التغير في سلسلة محارف الاستعلام هو الطريق الوحيد لخبء نسخ متباينة من صفحة ما.

تمكن ASP.NET من إنشاء إجرائية خاصة مهمتها تحديد الحاجة إلى خبء نسخة جديدة من الصفحة.

تستخدم لهذا الغرض الخاصة VaryByCustom.

أحد الاستخدامات الممكنة لهذا النمط من التحكم المخصص بالخبء هو تمكين خبء نسخة جديدة من الصفحة عند اختلاف المستعرض المستعمل.

يجب أن يضاف الرماز التالي إلى أعلى الصفحة المراد خبء نسخ منها:

```
<%@ OutputCache Duration="10" VaryByParam="None" VaryByCustom="Browser" %>
```

تستلزم المرحلة التالية كتابة النص البرمجي للإجرائية المسؤولة عن تحديد خبء نسخة جديدة.

يوضع الرماز التالي ضمن ملف التطبيق Global.asax:

```
public override string GetVaryByCustomString(
HttpContext context, string arg)
{
// Check for the requested type of caching.
if (arg == "browser")
{
// Determine the current browser.
string browserName;
browserName = Context.Request.Browser.Browser;
// Indicate that this string should be used to vary caching.
return browserName;
}
else
{
// For any other type of caching, use the default logic.
return base.GetVaryByCustomString(context, arg);
}
```

الخبء الجزئي:

يكتشف المطور في بعض الأحيان أنه لن يكون بإمكانه خبء كامل الصفحة لذلك يلجأ إلى خبء الجزء ذو التعقيد الأعلى والأكثر كلفة من حيث زمن البناء و التخزين و التي لا تتغير بشكل كبير نسبياً.

لنتمكن من استخدام الخبء الجزئي علينا بناء عنصر تحكم مخصص يحتوي الجزء المراد خبئه من الصفحة. يمكن بعد هذا إضافة الموجه OutputCache إلى عنصر التحكم المخصص.

لا يختلف مفهوم الخبء الجزئي عن المفهوم العام لخبء الصفحة بفرق واحد هو أن الصفحة لن تستطيع التفاعل مع محتوى عنصر التحكم الذي تمت استعادته من ذاكرة الخبء.

فإذا وفر عنصر التحكم الذي تم خبئه على سبيل المثال مجموعة من الخصائص، لن يتمكن رماز الصفحة من الوصول إلى، أو تعديل، هذه الخصائص.

ستتم إضافة كتلة HTML المخزنة كما هي ضمن الصفحة.

إعدادات الخبء

أحد المشاكل الأساسية لخبء الخرج هي ضرورة تضمين التعليمات ضمن الجزء الحاوي على التأشيرات في ملفات aspx أو الرماز الخاص بالصف.

مع أنّ هذه الطريقة ليست سيئة، تصبح المشكلة أكثر تعقيداً في حال كان عدد الصفحات التي يتم تفعيل الخبء فيها كبيراً.

عندها تصبح عملية تغيير الإعدادات الخاصة بالخبء في كل الصفحات عملية مرهقة.

تتضمن ASP.NET ميزة تدعى cache profile. تمكن هذه الخاصة من تعريف إعدادات الخبء لمجموعة من الصفحات ضمن ملف web.config ، عندها يمكن ربط هذه الإعدادات مع عدد كبير من الصفحات.

يكفي لتعديل إعدادات الخبء لجميع الصفحات تعديل الإعدادات من ضمن ملف web.config.

لتعريف إعدادات خبء نقوم باستخدام التأشيرة <add> ضمن القسم <outputCacheProfiles> كما يوضح المثال التالي:

```
<configuration>
<system.web>
<caching>
<outputCacheSettings>
<outputCacheProfiles>
<add name="ProductItemCacheProfile" duration="60" />
</outputCacheProfiles>
</outputCacheSettings>
</caching>
...
</system.web>
</configuration>
```

يمكننا استخدام الإعدادات التي تم تعريفها في أحد تشكيلات إعدادات الخبء ضمن صفحة محددة باستخدام الواصفة CacheProfile:

```
<% OutputCache CacheProfile="ProductItemCacheProfile" VaryByParm="None" %>
```

لتحديد تفاصيل أخرى متميزة لإحدى الصفحات يمكننا استخدام الموجه OutputCache أو استخدام التأشيرة <add> لتعريفه ضمن تشكيل مختلف.

خبء البيانات

يعتبر هذا الشكل من أشكال الخبء أكثر الأشكال مرونة ، لكنه يتطلب خطوات إضافية ضمن النص البرمجي.

الفكرة الأساسية في خبء البيانات أن يقوم المطور بشكل صريح بإضافة القيم والأغراض التي تعتبر عمليات إنشائها، تخزينها مكلفة إلى غرض مجموعة مبيت يدعى Cache.

الغرض Cache هو أحد خصائص صف Page. حيث تعيد هذه الخاصة مثيل من الصف System.Web.Caching.Cache.

يعمل هذا الغرض بشكل مشابه لغرض التطبيق Application الذي قمنا باستعراض عمله ضمن الجلسة الخاصة بإدارة حالة التطبيق.

يختلف غرض Cache عن غرض Application بثلاث نقاط أساسية:

- يدعم غرض Cache الطلبات المتزامنة: بمعنى أنه لا ضرورة لعملية إقفال وإلغاء إقفال الغرض Cache بشكل صريح عند إضافة، إزالة أو تعديل عنصر.
- يتم إزالة العناصر من ذاكرة الخبء بشكل آلي.
- دعم العناصر المرتبطة: أي هناك إمكانية لربط العناصر في ذاكرة الخبء بملف، جدول في قاعدة البيانات أو أي نمط آخر من الموارد. في حال تعديل أي من الموارد المرتبطة ستعتبر العناصر المخزنة في ذاكرة الخبء والمرتبطة به منتهية الصلاحية.

إضافة عناصر إلى غرض Cache:

يمكن إدراج عنصر ضمن غرض Cache بعدة طرق.

يمكن ببساطة استخدام اسم مفتاح جديد بالشكل:

```
Cache["KeyName"] = objectToCache;
```

هذه الطريقة غير محببة كونها لا تقدم أي تحكم بالزمن الذي سيتم ستبقى فيه المعلومات ضمن ذاكرة الخبء.

أحد الطرق الأخرى الأفضل للوصول إلى نفس النتيجة استخدام الطريقة Insert() كما يلي:

```
Cache.Insert(key, item, dependencies, absoluteExpiration, slidingExpiration);
```

يوضح الجدول التالي المعاملات التي تستخدمها هذه الطريقة:

المعامل	الوصف
key	سلسلة محرفية تحدد اسم العنصر الذي تم خبئه
item	الغرض المراد تخزينه ضمن ذاكرة الخبء
Dependencies	غرض من الصف CacheDependency يسمح بإنشاء ارتباط مع مورد ما.يجب إسناد القيمة null إلى هذه الخاصة في حال عدم وجود موارد مرتبطة.
absoluteExpiration	غرض DateTime يمثل تاريخ ووقت إزالة العنصر المحدد من ذاكرة الخبء.
slidingExpireation	الفترة بين طلبين على نفس العنصر قبل أن تقوم ASP.NET بإزالته من ذاكرة الخبء.القيمة الأفضلية لهذا المعامل هي 20

أمثلة عن استخدام هذه المعاملات هي:

```
Cache.Insert("MyItem", obj, null, DateTime.Now.AddMinutes(60), TimeSpan.Zero);

Cache.Insert("MyItem", obj, null, DateTime.MaxValue, TimeSpan.FromMinutes(10));
```

مثال بسيط

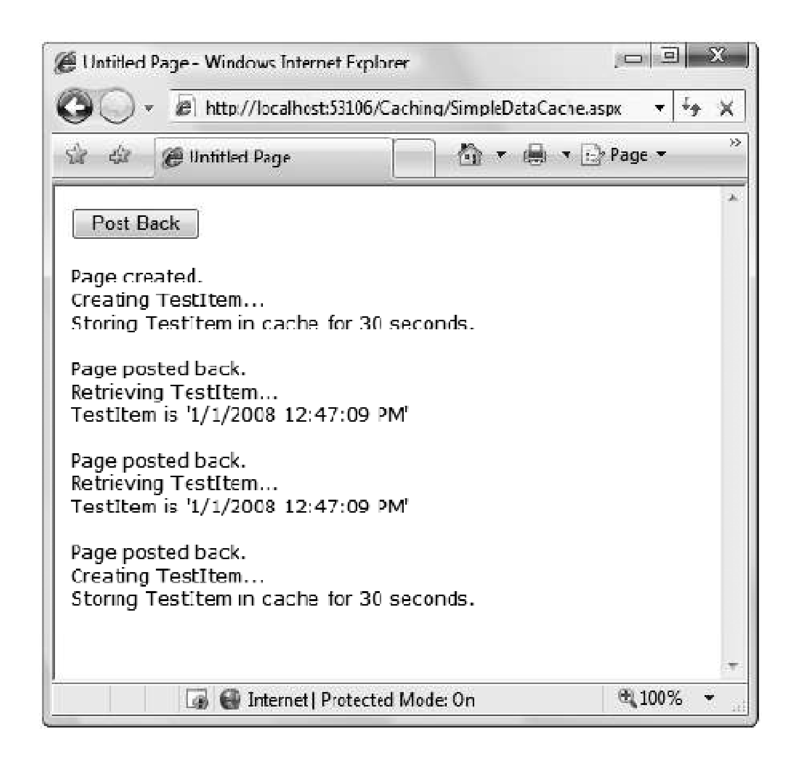
المثال التالي يوضح كيفية استخدام الطريقة Insert لخبء العنصر testItem.

يتم تنفيذ الرماز في الصفحة في كل مرة لأن الخبء محصور بعنصر معين ولا يشمل الصفحة كلها.

```
public partial class SimpleDataCache : System.Web.UI.Page
{
protected void Page_Load(Object sender, EventArgs e)
{
if (this.IsPostBack)
{
lblInfo.Text += "Page posted back.<br />";
}
else
{
lblInfo.Text += "Page created.<br />";
}
if (Cache["TestItem"] == null)
{
lblInfo.Text += "Creating TestItem...<br />";
```

```
DateTime testItem = DateTime.Now;
lblInfo.Text += "Storing TestItem in cache ";
lblInfo.Text += "for 30 seconds.<br />";
Cache.Insert("TestItem", testItem, null,
DateTime.Now.AddSeconds(30), TimeSpan.Zero);
}
else
{
lblInfo.Text += "Retrieving TestItem...<br />";
DateTime testItem = (DateTime)Cache["TestItem"];
lblInfo.Text += "TestItem is '" + testItem.ToString();
lblInfo.Text += "'<br />";
}
lblInfo.Text += "<br />";
}
}
```

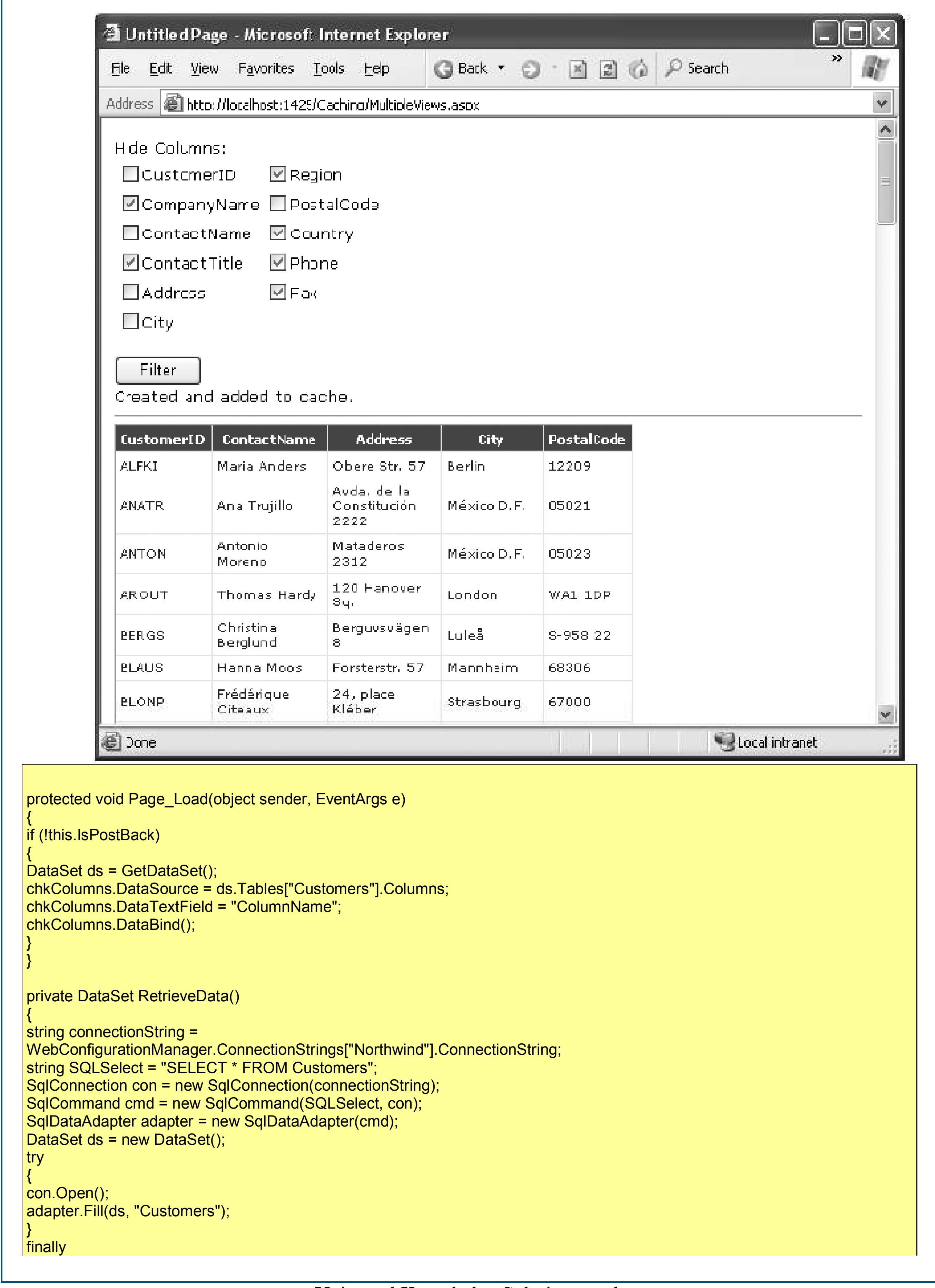
يمثل الشكل التالي خرج التطبيق عند النقر على زر post back.



الخبء بغرض توفير أكثر من مشهد

يتم في المثال التالي تخزين غرض DataSet ضمن ذاكرة الخبء.

يوفر هذا الغرض مصدر البيانات لعنصر تحكم GridView. باعتماد خبء غرض Dataset عوضاً عن خبء عدد كبير من المشاهد التي تنتج عن عملية تخصيص الغرض GridView كعرض أعمدة محددة فقط.




```
protected void Page_Load(object sender, EventArgs e)
{
if (!this.IsPostBack)
{
DataSet ds = GetDataSet();
chkColumns.DataSource = ds.Tables["Customers"].Columns;
chkColumns.DataTextField = "ColumnName";
chkColumns.DataBind();
}
}

private DataSet RetrieveData()
{
string connectionString =
WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
string SQLSelect = "SELECT * FROM Customers";
SqlConnection con = new SqlConnection(connectionString);
SqlCommand cmd = new SqlCommand(SQLSelect, con);
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
try
{
con.Open();
adapter.Fill(ds, "Customers");
}
finally
```

```
{
con.Close();
}
return ds;
}

{
con.Close();
}
return ds;
}

private DataSet GetDataSet()
{
DataSet ds = (DataSet)Cache["Customers"];
// Contact the database if necessary.
if (ds == null)
{
ds = RetrieveData();
Cache.Insert("Customers", ds, null, DateTime.MaxValue,
TimeSpan.FromMinutes(2));
lblCacheStatus.Text = "Created and added to cache.";
}
else
{
lblCacheStatus.Text = "Retrieved from cache.";
}
return ds;
}
```

الخبء باستخدام عنصر تحكم مصادر البيانات

تدعم عناصر تحكم مصادر البيانات مثل SlDataSource، ObjectDataSource، XmlDataSource ميزة خبء البيانات المبيتة. ينصح باستخدام الخبء بهذه الطريقة لأن عملية الاستعلام من قاعدة البيانات تتكرر مع كل إعادة إرسال للصفحة كذلك يمكن أن تتم إعادة الاستعلام لكل عنصر مرتبط بمصدر البيانات.

أي إذا كان لدينا ثلاثة عناصر تحكم مرتبطة بمصدر بيانات وحيد سينفذ الاستعلام ثلاث مرات. لذلك فإن تفعيل عملية الخبء في سيناريو مماثل سيؤدي إلى تحسن كبير في الأداء.

توفر عناصر تحكم مصادر البيانات مجموعة من الخصائص لدعم ميزة الخبء:

الخاصة	الوصف
EnableCaching	لتفعيل الخبء في عنصر تحكم مصدر البيانات يجب إسناد القيمة true إلى هذه الخاصة
CacheExpirationPolicy	تستخدم هذه الخاصة الترقيم DataSourceCacheExpiry. Absolute أو Sliding
CacheKeyDependency and sqlCacheDependency	تسمح هذه الخاصة بخبء عناصر مرتبطة بعناصر أخرى ضمن خبء البيانات

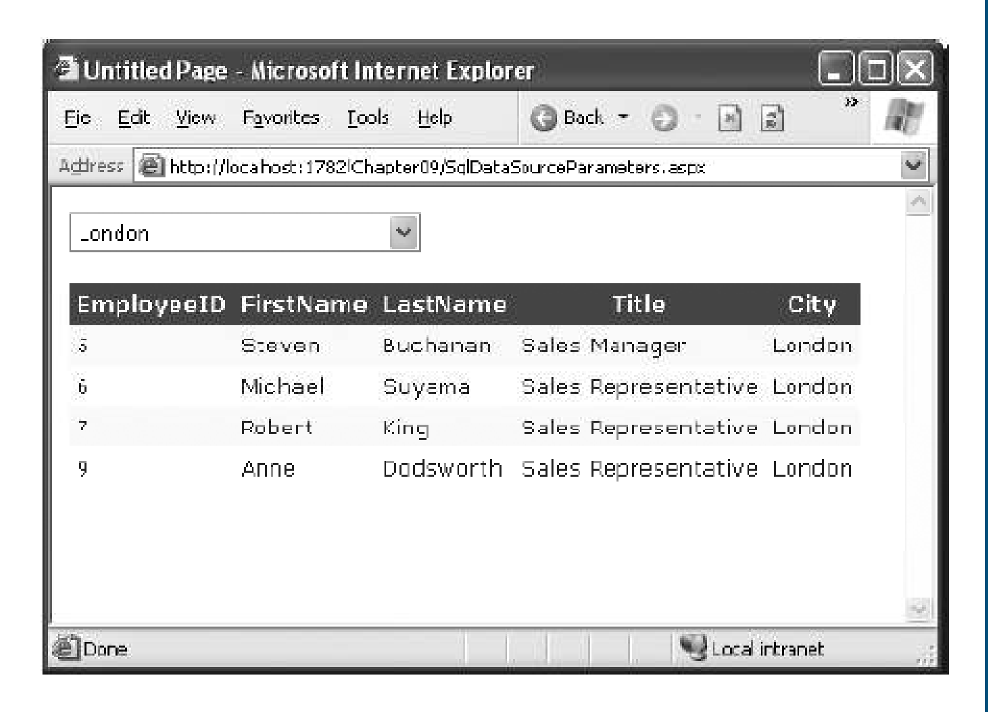
مثال عن الخبء في SqlDataSource

عند تفعيل الخبء في عنصر تحكم SqlDataSource تتم عملية خبء لنتائج SelectCommand.

عند استخدام المعاملات (Parameters) في إنشاء استعلام يتم خبء نتائج منفصلة لكل مجموعة من قيم المعاملات.

مثلاً لنفرض أننا نريد إنشاء صفحة تسمح باستعراض معلومات الموظفين بحسب المدينة، حيث يتم اختيار المدينة من قائمة منسدلة.

يتم استخدام عنصر التحكم SqlDataSource لملء سجلات الموظفين ضمن عنصر تحكم GridView كما هو موضح بالشكل:



نستخدم في هذا المثال عنصري تحكم مصدر قواعد بيانات يقوم أحدها بالاستعلام عن المدن و قيمه لا تتغير بتواتر كبير.

أما الآخر وهو عنصر تحكم GridView فيقوم بإظهار قائمة بالموظفين ومعلوماتهم:

```
<asp:SqlDataSource ID="sourceEmployeeCities" runat="server"
ProviderName="System.Data.SqlClient"
EnableCaching="True" CacheDuration="3600"
ConnectionString="<%$ ConnectionStrings:Northwind %>"
SelectCommand="SELECT DISTINCT City FROM Employees">
</asp:SqlDataSource>
<asp:DropDownList ID="lstCities" runat="server"
DataSourceID="sourceEmployeeCities"
DataTextField="City" AutoPostBack="True">
</asp:DropDownList>


<asp:SqlDataSource ID="sourceEmployees" runat="server"
ProviderName="System.Data.SqlClient"
EnableCaching="True" CacheDuration="600"
ConnectionString="<%$ ConnectionStrings:Northwind %>"
SelectCommand="SELECT EmployeeID, FirstName, LastName, Title, City
FROM Employees WHERE City=@City">
<SelectParameters>
<asp:ControlParameter ControlID="lstCities" Name="City"
PropertyName="SelectedValue" />
```

```
</SelectParameters>
</asp:SqlDataSource>
<asp:GridView ID="GridView1" runat="server"
DataSourceID="sourceEmployees" ... >
...
</asp:GridView>
```

الخبء مع وجود موارد مرتبطة:

قد يتغير محتوى مصدر البيانات مع مرور الزمن، وقد يتم استعمال نسخة قديمة من المعلومات دون معرفة.

تدعم ASP.NET ما يسمى الخبء المرتبط. يمكّن هذا النمط من الخبء من ربط عنصر ضمن ذاكرة الخبء مع أحد الموارد بحيث يتم إزالة النسخة المخبأة واعتبارها منتهية الصلاحية عند أي تغيير على هذا المورد.

توفر ASP.NET ثلاثة أنواع من الخبء المرتبط:

1- الربط مع عنصر آخر ضمن ذاكرة الخبء
2- الربط مع ملف أو مجلد
3- الربط مع استعلام من قاعدة البيانات

الارتباط بالملفات

لاستخدام الخبء المرتبط نستعمل الغرض CacheDependency.

على سبيل المثال يقوم الرماز التالي بإنشاء ارتباط مع ملف XML باسم ProductList.xml.

سيتم إنهاء صلاحية العنصر المرتبط المخزن في ذاكرة الخبء وإخلائه من الذاكرة عند حصول أي تغير في الملف السابق:

```
// Create a dependency for the ProductList.xml file.
CacheDependency prodDependency = new CacheDependency(
Server.MapPath("ProductList.xml"));
// Add a cache item that will be dependent on this file.
Cache.Insert("ProductInfo", prodInfo, prodDependency);
```

الخبء المرتبط بعنصر آخر ضمن ذاكرة الخبء

يوفر الغرض CacheDependency مشيد يقبل مصفوفة من أسماء الملفات ومصفوفة من مفاتيح الخبء.

يمكن باستخدام مصفوفة مفاتيح الخبء إنشاء ارتباط بين عنصر ضمن ذاكرة الخبء وعنصر آخر مخزن ضمن هذه الذاكرة.

فيما يلي مثال يقوم بربط عنصر ضمن ذاكرة الخبء مع عنصر آخر دون الربط مع ملف:

```
Cache["Key1"] = "Cache Item 1";
// Make Cache["Key2"] dependent on Cache["Key1"].
string[] dependencyKey = new string[1];
dependencyKey[0] = "Key1";
CacheDependency dependency = new CacheDependency(null, dependencyKey);
Cache.Insert("Key2", "Cache Item 2", dependency);
```

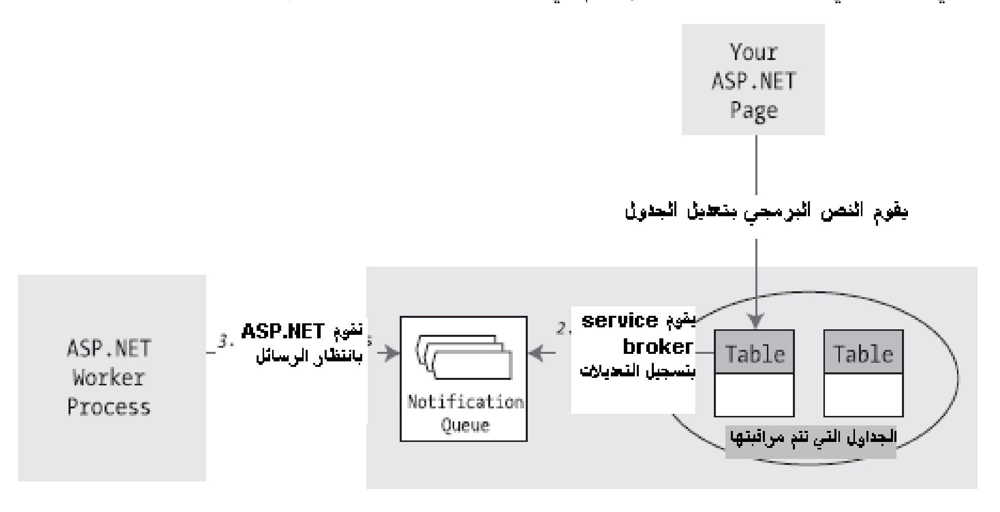
ملاحظة: قمنا باستخدام القيمة null لمعامل مصفوفة الملفات المرتبطة ضمن مشيد CacheDependency.

الخبء المرتبط بقاعدة بيانات SQL Server 2005 and 2008

توفر قاعدة بيانات SQL Server إمكانية تسجيل استعلام ما لتتم مراقبته من قبل SQLServer بحيث يتم إرسال رسالة إعلام إلى إجرائية ASP.NET العاملة لإبلاغها بحصول تغيير.

- يقوم SQLServer باعتبار غرض البيانات لديك منتهي الصلاحية عند حصول تغيير على صف من الصفوف التي يعيدها الاستعلام المسجل.
- يتم إرسال رسالة الإعلام بالتغيير لأول مرة يحصل فيها التغيير و ليس مع كل تغيير لاحق حيث تعتبر هذه الطريقة أكثر ذكاء حيث تتجنب إرسال رسائل سيتم إهمالها لاحقاً بعد الأخذ بأول رسالة.
- لا حاجة إلى إعدادات خاصة لتفعيل الإعلام في SQL server 2005 ,2008 (كما هي الحال في نسخ سابقة) لكن يكفي استخدام الطريقة المشتركة SqlDependency.Start() ضمن الرماز.

يوضح الشكل التالي الآلية التي تعمل فيها رسائل الإعلام في SQL Server 2005, 2008



تعمل رسائل الإعلام مع استعلامات SELECT ومع الإجرائيات المخزنة.

لضمان دعم استعلامات SELECT لابد من مراعاة القواعد التالية:

لا بد من استخدام الاسم الكامل للجداول بالصيغة [Owner].table أي استخدام الاسم dbo.Employees عوضاً عن استخدام Employees فقط.

لا يمكن للاستعلامات أن تتضمن توابع تجميعية مثل COUNT()، MAX()، MIN() أو AVERAGE()

لا يمكن استخدام إشارة * ضمن الاستعلام لتحديد جميع الأعمدة بل يجب ذكر أسمائها بشكل صريح. أي أن الصيغة Select * from table غير قانونية.

تفعيل سمسار الخدمات

لا يتم تفعيل سمسار الخدمات في SQL Server2005-2008 عند تثبيته لذلك يجب تفعيله بشكل صريح لكل قاعدة بيانات.

نقوم أولاً بتشغيل موجه الأوامر من Visual Studio:

Programs >Microsoft Visual Studio 2008 >Visual Studio Tools >Visual Studio 2008 Command Prompt

بعدها نقوم بتشغيل الأمر SqlCmd.exe مع استخدام المعامل S– و اسم المخدم

```
SqlCmd -S localhost\SQLEXPRESS
```

المرحلة التالية هي تفعيل سمسار الخدمات على قاعدة البيانات المراد التعامل معها وذلك عبر استخدام SqlCmd.exe لتشغيل هذه العبارة:

```
USE Northwind
ALTER DATABASE Northwind SET ENABLE_BROKER
GO
```

تأهيل خدمة الخبء

كما ذكرنا سابقاً لا بد من تأهيل خدمة الخبء باستخدام الطريقة SqlDependency.Start() قبل الشروع باستخدامها.

يقوم الرماز التالي مثلاً بتفعيل خدمة التنصت على مخدم الوب لاستلام رسائل الإعلام بالتغيرات:

```
string connectionString = WebConfigurationManager.ConnectionStrings[
"Northwind"].ConnectionString;
SqlDependency.Start(connectionString)
```

لا حاجة لإجراء هذه العملية أكثر من مرة ضمن كامل فترة حياة التطبيق، لذلك فمن المنطقي وضع هذه الطريقة ضمن مقبض حدث بدء عمل التطبيق Application_Start() في الملف Golbal.asax.

إنشاء خبء الربط

لا بد لإنشاء غرض مرتبط باستعلام من تحديد الأمر الذي سيستخدم لاستعادة البيانات ليتمكن المخدم من تحديد الصفوف التي ستتم مراقبتها.

نقوم لهذا الغرض بإنشاء غرض SqlCacheDependency الذي يقبل معامل يحدد الاستعلام المطلوب من النمط SqlCommand.

يوضح المثال التالي كيفية تطبيق هذه الخطوات:

```
// Create the ADO.NET objects.
SqlConnection con = new SqlConnection(connectionString);
string query =
"SELECT EmployeeID, FirstName, LastName, City FROM dbo.Employees";
SqlCommand cmd = new SqlCommand(query, con);
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
// Fill the DataSet.
DataSet ds = new DataSet();
adapter.Fill(ds, "Employees");
// Create the dependency.
SqlCacheDependency empDependency = new SqlCacheDependency(cmd);
// Add a cache item that will be invalidated if one of its records changes
// (or a new record is added in the same range).
```

```
Cache.Insert("Employees", ds, empDependency);
```

سيؤدي أي تعديل على صفوف الجدول Employee إلى إرسال إعلام من قبل قاعدة البيانات.

يتم استقبال هذا الإعلام من إجرائية ASP.NET العاملة وإنهاء صلاحية غرض DataSet و حذفه من ذاكرة الخبء.

لا بد في المرة التالية ،التي تتم فيها إعادة إنشاء هذا الغرض بالوضع الجديد بعد التعديل، من إعادة إضافة الغرض من جديد إلى ذاكرة الخبء باستخدام غرض SqlCacheDependency جديد.

